Distributed Synthesis for Regular and Contextfree Specifications*

Wladimir Fridman and Bernd Puchala

RWTH Aachen University, Germany
{fridman@automata,puchala@logic}.rwth-aachen.de

Abstract. We adress the controller synthesis problem for distributed systems with regular and deterministic contextfree specifications. Our main result is a complete characterization of the decidable architectures for local specifications. This extends existing results on local specifications in two directions. First, we consider arbitrary, not necessarily acyclic, architectures and second, we allow deterministic contextfree specifications. Moreover, we show that for global deterministic contextfree specifications, even very simple architectures are undecidable.

1 Introduction

Open non-terminating reactive systems are computing systems which continously interact with an environment. Such systems are modeled as infinite games between a controller for the system and the environment. As the behavior of the environment is usually not constrained a priori in such settings, it is considered as being antagonistic. Non-terminating reactive systems have first been considered in the context of switching circuits. Church's Synthesis Problem [2] is to decide whether there exists a switching circuit such that all possible input/output behaviors of the circuit satisfy a given specification and, if such a circuit exists, it should be constructed effectively. The first solution to this problem has been given by Büchi and Landweber [1] who showed that for any specification formulated in monadic second order logic over words, the synthesis problem is decidable and finite state solutions can be constructed effectively.

Since then, non-terminating reactive systems have received growing attention in computer science. Such systems naturally capture many settings where a given plant should be controlled in such a way, that any constrained system behavior satisfies a certain specification. Moreover, the prospect of being able to construct such systems automatically from a given specification, rather than verifying a system that has already been built, has led to intensive research on the controller synthesis problem [9, 13, 15] and to extensions of the basic setting in various directions. For example, other specification formalisms have been considered like temporal logics [12] and contextfree specifications [17], the systems have been extended to distributed systems which consist of several components [14], and stochastic versions of reactive systems have been investigated [4].

* Full Version of [7].

We consider distributed systems with regular and contextfree specifications. Such a distributed system is specified by an architecture which consists of a set of processes and channels via which the processes can communicate. Distributed systems have first been considered in [14] where it has been shown that in general, the distributed controller synthesis problem is undecidable for specifications from the linear time temporal logic LTL. Moreover, for pipelines, a special class of acyclic architectures, decidability has been proved for LTL specifications. In [10] the decidability results have been extended to certain classes of architectures with cycles and to specifications from the branching time logic CTL. Finally, in [5], a full characterization of the decidable architectures has been given by means of certain patterns of information flow, called information forks: Two processes form an information fork if they are incomparably informed and the controller synthesis problem for an architecture is decidable if, and only if, it does not contain an information fork. This holds for both LTL and CTL specifications.

In [11], the concept of local specifications was introduced. There, any system process has an individual specification which defines the correct behaviors of just this process instead of the whole system. Obviously, any set of regular local specifications can be transformed into a regular global specification, so all decidability results for global regular specifications hold for local regular specifications as well. However, there are architectures, called two-flanked pipelines, which contain an information fork but are decidable for local regular specifications [11]. Moreover, for the class of acyclic architectures, a characterization of the decidable architectures was established for regular specifications.

We extend this result to architectures which may contain cycles and to specifications which are regular or deterministic contextfree. Notice that in the case of global specifiations, channels from processes with a lower level of information to better informed processes (backward channels) are futile, so architectures without information forks can be transformed into a normal form which does not have cycles [5]. These techniques do not, however, preserve local specifications and in fact, in the case of local specifications, backward-channels can be significant, as they may increase the access of the local specifications to the overall (global) system behavior. Therefore, to deal with cycles in the case of local specifications, one has to use different methods. Also, processes not reachable from the environment cannot be eliminated in general, so a decidable architecture may consist of several basic decidable subarchitectures possibly connected via such unreachable processes. Our analysis is centered around those two structural aspects and the higher expressive power of deterministic contextfree specifications. In Section 4 we first prove decidability and undecidability results for some special classes of architectures, followed by a complete characterization of the decidable architectures in Section 5.

In Section 3, we also show that as soon as one considers global deterministic contextfree specifications, even very simple architectures become undecidable. For architectures with only one system process, it has been shown that they are decidable for deterministic contextfree specifictions [17]. We show that this is

not the case for architectures with at least two system processes or at least one channel from the environment which cannot be read by any system process.

2 Preliminaries

The set of natural numbers is denoted by $\mathbb{N}$ and for a set X the power set of X is denoted by $\mathcal{P}(X)$. The boolean alphabet is denoted by $\mathbb{B} = \{\top, \bot\}$. Moreover, for any alphabet Σ and words $\alpha, \beta \in \Sigma^* \cup \Sigma^\omega$ and $n \in \mathbb{N}$ we write $\alpha(n)$ for the n-th letter of α, $\alpha{\restriction_n} = \alpha(0) \dots \alpha(n-1)$ and $\alpha \sqsubseteq \beta$ if α is a prefix of β. For a set A and a word $\alpha \in A^\omega$, we denote $\mathrm{Inf}(\alpha) = \{a \in A \mid \alpha(i) = a \text{ for infinitely many } i\}$.

For an integer $k > 0$ let $[k]$ denote the set $\{0, ..., k-1\}$. For a cartesian product $A = A_0 \times \ldots \times A_{n-1}$ and $I \subseteq [n]$ we denote $A_I = \prod_{i \in I} A_i$. Moreover, $\mathrm{Pr}_I(a) = (a_i)_{i \in I}$ for an element $a = (a_0, \dots, a_{n-1}) \in A$, $\mathrm{Pr}_I(\alpha) = \mathrm{Pr}_I(\alpha(0))\mathrm{Pr}_I(\alpha(1)) \dots$ for a word $\alpha \in A^* \cup A^\omega$ and $\mathrm{Pr}_I(L) = \{\mathrm{Pr}_I(\alpha) \mid \alpha \in L\}$ for a language $L \subseteq A^* \cup A^\omega$. However, usually we do not refer to an explicit ordering of the components of a cartesian product and write Pr_{A_I} instead of Pr_I. If A has certain identical components, this is not unambigious, but it will be clear from the context to which components the operator projects. Moreover, if $\alpha \in X^\omega$ and $\beta \in Y^\omega$, $\alpha^\frown\beta \in (X \times Y)^\omega$ denotes the ω-word with $(\alpha^\frown\beta)(i) = (\alpha(i), \beta(i))$ for all $i \in \mathbb{N}$.

For a function $\sigma\colon \Sigma^* \to \Sigma'$, we define the finite iteration $\sigma^*\colon \Sigma^* \to (\Sigma')^*$ of σ by $\sigma^*(u) = \sigma(u{\restriction_0})\sigma(u{\restriction_1}) \dots \sigma(u{\restriction_{|u|-1}})$ and the ω-iteration $\sigma^\omega\colon \Sigma^\omega \to (\Sigma')^\omega$ of σ by $\sigma^\omega(\alpha) = \sigma(\alpha{\restriction_0})\sigma(\alpha{\restriction_1}) \dots$. The ω-language which is generated by σ over a language $L_{\mathrm{in}} \subseteq \Sigma^\omega$ is $L^\omega(\sigma) = \{\sigma^\omega(\alpha) \mid \alpha \in L_{\mathrm{in}}\} \subseteq (\Sigma')^\omega$. Moreover, the language of finite words generated by σ over L_{in} (or, more precisely, over $\{u \in \Sigma^* \mid u \sqsubseteq \alpha \text{ for some } \alpha \in L_{\mathrm{in}}\}$) is $L^*(\sigma) = \{\sigma^*(u) \mid u \sqsubseteq \alpha \text{ for some } \alpha \in L_{in}\}$. Notice that $L^*(\sigma)$ coincides with the set of all $v \in (\Sigma')^*$ such that $v \sqsubseteq \beta$ for some $\beta \in L^\omega(\sigma)$. For any functions $f\colon A \to B$ and $g\colon B \to C$ the composition $f \circ g\colon A \to C$ is defined by $(f \circ g)(a) = g(f(a))$. Moreover, for $A' \subseteq A$, we denote $f(A') = \{f(a) \mid a \in A'\}$.

Architectures. An architecture $\mathfrak{A} = (P, C, r)$ consists of the following components. $P = \{p_{\mathrm{env}}\} \cup P_{\mathrm{sys}}$ is the set of processes where $p_{\mathrm{env}} = p_0$ takes the role of the environment and $P_{\mathrm{sys}} = \{p_1, \dots, p_n\}$ with $n \geq 1$ are system processes. Moreover, $C = \bigcup_{p \in P} C_p$ is the set of channels where the sets C_p are pairwise disjoint and $r\colon C \to P$ is a function, assigning for each channel a process which reads it such that $r(C_p) \subseteq P_{\mathrm{sys}}$ for all $p \in P_{\mathrm{sys}}$. We assume that, for all $p \in P_{\mathrm{sys}}$, $r^{-1}(p) \neq \emptyset$ and $C_p \neq \emptyset$, i.e., each system process has at least one input and one output channel. So basically, an architecture is a directed graph with multi-egdes.

An architecture $\mathfrak{A}$ is called connected, if P_{sys} induces a connected subgraph of $\mathfrak{A}$. Consider a set $Q \subseteq P_{\mathrm{sys}}$. If the subgraph of $\mathfrak{A}$ induced by $Q \cup \{p_{\mathrm{env}}\}$ is an architecture, then we denote this architecture by $\mathfrak{A}(Q)$ and we say that $\mathfrak{A}(Q)$ is the subarchitecture of $\mathfrak{A}$ induced by Q. (Notice that not each set $Q \subseteq P_{\mathrm{sys}}$ induces a subarchitecture of $\mathfrak{A}$.) An architecture $\mathfrak{A}'$ is a subarchitecture of $\mathfrak{A}$ if $\mathfrak{A}' = \mathfrak{A}(Q)$ for some set $Q \subseteq P_{\mathrm{sys}}$.

For $p \in P$, $H_p = \{c \in C_p | r(c) = p\}$ are called hidden channels of p, i.e., they cannot be read by any other process. C_{p_0} are external input channels and

Fig. 1. Pipeline and two-flanked pipeline with backward-channels

$H_{p_0} \subseteq C_{p_0}$ are hidden input channels. $\bigcup_{i=1}^{n} C_{p_i} \setminus H_{p_i}$ are internal communication channels and the channels $\bigcup_{i=1}^{n} H_{p_i}$ are used to model external output channels.

We say that process p sends information to process $p' \neq p$ if there is some channel $c \in C_p$ such that $p' = r(c)$. Process p is called reachable, if there is a directed path from p_{env} to p. Process p is better informed than $p' \neq p$ if p is reachable and each directed path from p_{env} to p' goes through p. Notice that a process may send information to another process via multiple channels. However, if more convenient, we can assume w.l.o.g. that there is at most one such channel.

An architecture $\mathfrak{A} = (P, C, r)$ is called pipeline (two-flanked pipeline) with backward-channels if $r(C_{p_0}) = \{p_1\}$ ($r(C_{p_0}) = \{p_1, p_n\}$ in case of a two-flanked pipeline) and, for $i \in [n] \setminus \{0\}$, $r(C_{p_i}) \subseteq \{p_j \mid 0 < j \leq i+1\}$ (see Figure 1). A channel $c \in C_{p_i}$ with $r(c) = p_j$ is called backward-channel if $0 < j < i$ and it is called forward-channel if $j = i + 1$. Moreover, $\mathfrak{A}$ is called (two-flanked) pipeline if it has no backward-channels.

A labeling $(\Sigma_c)_{c \in C}$ for $\mathfrak{A}$ assigns to any channel $c \in C$ a nonempty finite set Σ_c of signals which can be sent along c. We define, for every $p \in P$, the input and output alphabets of p as $\Sigma^p_{\text{in}} = \prod_{c \in r^{-1}(p)} \Sigma_c$ and $\Sigma^p_{\text{out}} = \prod_{c \in C_p} \Sigma_c$. The local alphabet of p is $\Sigma^p = \Sigma^p_{\text{in}} \times \Sigma^p_{\text{out}}$. The global system alphabet is defined as $\Sigma^{\mathfrak{A}} = \prod_{c \in C} \Sigma_c = \prod_{i=0}^{n} \Sigma^{p_i}_{\text{out}}$. At each point in time i every process p writes a letter $\alpha_p(i) \in \Sigma^p_{\text{out}}$ to the corresponding channels $c \in C_p$. A global system behavior is an ω-word $\alpha = \alpha_{p_0} \frown \ldots \frown \alpha_{p_n}$ from $(\Sigma^{\mathfrak{A}})^\omega$. For $p \in P_{\text{sys}}$, the local process behavior of p is $\beta_p \frown \alpha_p$, where $\beta_p = \Pr_{\Sigma^p_{\text{in}}}(\alpha)$,

A global system specification is a language $L \subseteq (\Sigma^{\mathfrak{A}})^\omega$ consisting of all correct system behaviors. A local specification for process p is a language $L_p \subseteq (\Sigma^p)^\omega$. For a collection $(L_{p_1}, \ldots, L_{p_n})$ of local specifications for the system processes, the corresponding global system specification is the language $L \subseteq (\Sigma^{\mathfrak{A}})^\omega$ such that $\Pr_{\Sigma^p}(L) = L_p$ for any system process $p \in P_{\text{sys}}$.

A local strategy for process p maps a local input history of process p to the next output symbol of process p, i. e., it is a function $\sigma_p \colon (\Sigma^p_{\text{in}})^* \to \Sigma^p_{\text{out}}$. The local behavior $\beta_p \frown \alpha_p$ of process p is consistent with σ_p, if $\alpha_p(i) = \sigma_p(\beta_p{\restriction}_i)$ for all $i \in \mathbb{N}$. For a language $L_{\text{in}} \subseteq (\Sigma^p_{\text{in}})^\omega$, the local strategy σ_p is called winning on L_{in} if any local behavior $\beta_p \frown \alpha_p$ of p with $\beta_p \in L_{\text{in}}$ which is consistent with σ_p is in L_p. It is called winning for process p, if it is winning on $(\Sigma^p_{\text{in}})^\omega$.

A joint strategy for $p_1, \ldots, p_n$ is a tuple $\sigma = (\sigma_{p_1}, \ldots, \sigma_{p_n})$ where each σ_{p_i} is a local strategy for process p_i. A global system behavior $\alpha = \alpha_{p_0} \frown \ldots \frown \alpha_{p_n}$ is consistent with σ, if the local process behavior of each system process p is consistent with σ_p. The strategy σ is winning, if any system behavior which is

consistent with σ is in the global system specification L. Notice that a joint strategy which consists of local winning strategies is also winning for L. The converse is, however, not true in general: a local strategy for a process p which is part of a joint winning strategy is not necessarily locally winning as the inputs that p receives from other system processes are constrained by their local strategies.

A specification $L \subseteq (\Sigma^{\mathfrak{A}})^\omega$ is realizable in an architecture $\mathfrak{A}$ with labeling $(\Sigma_c)_{c\in C}$ if there is a joint winning strategy for processes $p_1, \ldots, p_n$. The realizability problem is to decide, given an architecture $\mathfrak{A}$, a labeling $(\Sigma_c)_{c\in C}$ and a specification $L \subseteq (\Sigma^{\mathfrak{A}})^\omega$, whether L is realizable in $\mathfrak{A}$. For a class $\mathcal{L}$ of specifications we say that an architecture $\mathfrak{A}$ is decidable for specifications from $\mathcal{L}$ if the realizability problem is decidable for the fixed architecture $\mathfrak{A}$ when specifications are restricted to $\mathcal{L}$.

Specifications. We consider regular and deterministic contextfree specifications. Regular specifications are those which can be recognized by parity automata. Notice that deterministic, nondeterministic and alternating parity automata over words have all the same expressive power [16].

Deterministic contextfree specifications are those which can be recognized by deterministic parity pushdown automata (parity DPDA) [3], i.e., finite state automata which additionally have access to a stack-memory. We also consider deterministic 1-counter specifications, i.e., recognizable by parity DPDA with only a single stack-symbol. Notice that deterministic contextfree languages form a proper subclass of contexfree languages and that, while games with deterministic contextfree winning condition are decidable [17], nondeterministic contextfree games are undecidable (see, e.g., [6]). In the following, we fix our notation for pushdown automata.

Pushdown Automata. A parity pushdown automaton has the form $\mathcal{P} = (Q, \Sigma, \Gamma, q_{\text{in}}, \delta, \bot, \text{col})$, where Q is the finite set of states with initial state q_{in}, Σ is the input alphabet, Γ is the pushdown alpahbet with initial stack symbol $\bot \notin \Gamma$ and $\text{col}\colon Q \to [k]$ is a coloring function for some $k \in \mathbb{N}$. Moreover, $\delta\colon Q \times (\Sigma_\epsilon) \times \Gamma_\bot \to \mathcal{P}(Q \times \Gamma_\bot^*)$ is the transition function, where $\Gamma_\bot = \Gamma \cup \{\bot\}$ and $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and we require that the initial stack symbol $\bot$ can neither be deleted from nor written to the stack, that means, for any $(q, a, \bot) \in Q \times \Sigma_\epsilon \times \bot$ we have $\delta(q, a, \bot)$ consists only of tuples of the form $(q', \gamma\bot)$ for some $q' \in Q$ and $\gamma \in \Gamma^*$. We call $\mathcal{P}$ deterministic, if for all $a \in \Sigma$, all $A \in \Gamma_\bot$ and all $q \in Q$ we have $|\delta(q, a, A)| + |\delta(q, \epsilon, A)| \leq 1$.

A configuration of $\mathcal{P}$ is a tuple $C = (q, \gamma) \in Q \times \Gamma^*\bot$. For $a \in \Sigma_\epsilon$ we write $(q, A\gamma) \overset{a}{\vdash} (q', \gamma'\gamma)$ if $(q', \gamma') \in \delta(q, a, A)$. A run ρ of $\mathcal{P}$ on a word $\alpha \in \Sigma^\omega$ is a sequence $C_0C_1\ldots$ of configurations of $\mathcal{P}$ such that $C_0 = (q_{\text{in}}, \bot)$ and for any $i \in \mathbb{N}$ we have $C_i \overset{\beta(i)}{\vdash} C_{i+1}$ where $\beta \in \Sigma_\epsilon^\omega$ is some word such that α is obtained from β by deleting all symbols ϵ.

The run ρ is accepting, if $\min[\text{col}(\text{Inf}(\text{Pr}_Q(\rho)))]$ is even. The language recognized by $\mathcal{P}$ is $L(\mathcal{P}) = \{\alpha \in \Sigma^\omega \mid \text{ there is an accepting run of } \mathcal{P} \text{ on } \alpha\}$. A language $L \subseteq \Sigma^\omega$ is called (deterministic) contextfree, if there is a (determinis-

tic) parity pushdown automaton $\mathcal{P}$ which recognizes L. A pushdown automaton $\mathcal{P}$ is called 1-counter if $|\Gamma| = 1$.

Trees and Tree-Automata. For a set X, an X-tree is a prefix closed set $T \subseteq X^*$. It is called full, if $T = X^*$. For an alphabet Σ, a Σ-labeled X-tree is a function $t\colon T \to \Sigma$ for some X-tree T. The tree t is called full if T is full. Unless explicitly mentioned otherwise, we consider only full trees here. By X_Σ we denote the set of all (full) Σ-labeled X-trees.

We use alternating parity tree automata (parity ATA) and nondeterministic parity pushdown tree automata (parity NPDTA) on such trees, where X is finite. Notice that for any parity ATA $\mathcal{A}$ there is an equivalent nondeterministic parity tree automaton (parity NTA) $\mathcal{N}$, that means, $L(\mathcal{A}) = L(\mathcal{N})$ [16]. Moreover, like for parity NTA, the nonemptiness problem for parity NPDTA is decidable [8]. In the following, we fix our notation for tree-automata.

An alternating pushdown tree automaton (APDTA) over Σ-labeled X-trees is given by a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, q_{\text{in}}, \delta, \bot, \text{acc})$ with a finite set Q of states, a transition function $\delta\colon Q \times \Sigma_\epsilon \times \Gamma_\bot \to B^+(\text{Dir}_{\circlearrowleft} \times Q \times \Gamma_\bot^*)$, where $B^+(\text{Dir}_{\circlearrowleft} \times Q \times \Gamma_\bot^*)$ is the set of all positive Boolean formulas over propositional variables from $\text{Dir}_{\circlearrowleft} \times Q \times \Gamma_\bot^*$ with $\text{Dir} = \{\downarrow_x \,|x \in X\}$ and $\text{Dir}_{\circlearrowleft} = \text{Dir} \cup \{\circlearrowleft_\epsilon\}$. Moreover, we have an acceptance component acc $\subseteq Q^\omega$. For Boolean formulas we assume, as usual, that $\wedge$ has precendence over $\vee$. Notice that with this definition, every formula from $B^+(\text{Dir}_{\circlearrowleft} \times Q \times \Gamma_\bot^*)$ is in disjunctive normalform. We denote such formulas ϕ in DNF also as sets $\phi = \{\psi_1, \ldots, \psi_k\}$ of conjuncts and we denote the conjuncts ψ also as sets $\psi \subseteq \text{Dir}_{\circlearrowleft} \times Q \times \Gamma_\bot^*$ of propositional variables. This allows us to use the notation $\psi \in \phi$ and $(d, q, \gamma) \in \psi$ unambigiously. As for pushdown word automata we assume that the $\bot$ symbol is neither written to nor deleted from the pushdown stack.

We consider parity tree automata where the acceptance component acc is given by a coloring function col : $Q \to [k]$ for some $k \in \mathbb{N}$ and Muller tree automata where acc is given by a collection $\mathcal{F}$ of subsets of Q. For parity tree automata, acc consists of those sequences $\alpha \in Q^\omega$ such that $\min[\text{col}(\text{Inf}(\alpha))]$ is even. For Muller tree automata, acc consists of those sequences $\alpha \in Q^\omega$ such that $\text{Inf}(\alpha) \in \mathcal{F}$.

A run of $\mathcal{A}$ on a Σ-labeled X-tree $t : X^* \to \Sigma$ is a *not* necessarily full Σ_r-labeled $\mathbb{N}$-tree $\rho : T \to \Sigma_r$ where $\Sigma_r = X^* \times Q \times \Gamma^* \bot$, such that the following conditions hold.

(1) $\epsilon \in T$ and $\rho(\epsilon) = (\epsilon, q_{\text{in}}, \bot)$.
(2) If $y \in T$ with $\rho(y) = (x, q, A\gamma)$ and $\delta(q, t(x), A) = \phi$ then there is some conjunct $\psi = \{(d_0, q_0, \gamma_0), \ldots, (d_{n-1}, q_{n-1}, \gamma_{n-1})\} \subseteq \text{Dir}_{\circlearrowleft} \times Q \times \Gamma^*$ in ϕ such that the set of successors of y in T is precisely $\{y \cdot i \,|\, i = 0, \ldots, n-1\}$ and $\rho(y \cdot i) = (x \cdot d_i, q_i, \gamma_i\gamma)$ where $x \cdot d = x \cdot z$ if $d = \downarrow_z$ and $x \cdot d = x$ if $d = \circlearrowleft_\epsilon$.

The run ρ is called accepting, if for each infinite path π through T, $\text{Pr}_Q(\pi) \in$ acc. For such a run ρ, we also sometimes refer to the X-component of $\rho(y)$ for

some $y \in T$ by which we mean the last symbol of the X^*-component of $\rho(y)$, i.e., $x(|x|-1)$ where $\rho(y) = (x, q, A\gamma)$.

The automaton $\mathcal{A}$ accepts a Σ-labeled X-tree t, if there is an accepting run of $\mathcal{A}$ on t. The language of $\mathcal{A}$ is $L(\mathcal{A}) = \{t \in X_\Sigma \mid \mathcal{A} \text{ accepts } t\}$.

Now w.l.o.g. let $X = [k]$ for some $k \in \mathbb{N}$. The automaton $\mathcal{A}$ is called nondeterministic (NPDTA) if, for each $(q, A) \in Q \times \Gamma_\perp$, either for all $a \in \Sigma$ there is some $(q', \gamma) \in Q \times \Gamma^*$ such that $\delta(q, a, A) = (\circlearrowleft_\epsilon, q', \gamma)$ or, for all $a \in \Sigma$, $\delta(q, a, A)$ has the form

$$\bigvee_{j=0}^{n-1} (\downarrow_0, q_0^j, \gamma_0^j) \wedge \ldots \wedge (\downarrow_{k-1}, q_{k-1}^j, \gamma_{k-1}^j)$$

for some $n \in \mathbb{N}$.

An alternating (nondeterministic) tree automaton, ATA (NTA) for short, is an APDTA (NPDTA) where $\delta : Q \times \Sigma \times \{\perp\} \to B^+(\mathrm{Dir} \times Q \times \{\perp\})$. We usually omit the $\perp$-symbol in the description of such an automaton and regard δ as a function $\delta : Q \times \Sigma \to B^+(\mathrm{Dir} \times Q)$.

Widening. The widening operator $\mathrm{wide}(t, Y)$ on a Σ-labeled X-tree t yields the Σ-labeled $X \times Y$-tree $t' = \mathrm{wide}(t, Y)$ with $t'(x, y) = t(x)$. For any parity NTA $\mathcal{A}$ over Σ-labeled $X \times Y$-trees, there is a parity NTA $\mathcal{B}$ over Σ-labeled X-trees, which accepts a tree t if, and only if, $\mathrm{wide}(t, Y) \in L(\mathcal{A})$ [9].

Product of Automata. Usually, given two automata $\mathcal{A}$ and $\mathcal{B}$ of the same kind and over the same objects, there is a canonical notion of the product of $\mathcal{A}$ and $\mathcal{B}$, denoted $\mathcal{A} \times \mathcal{B}$, such that the language of $\mathcal{A} \times \mathcal{B}$ is the intersection of the languages of $\mathcal{A}$ and $\mathcal{B}$. Here, we will need the (less standard) product of nondeterministic parity tree automata and nondeterministic parity pushdown tree automata which we define as follows. Given a parity NTA $\mathcal{A} = (Q^\mathcal{A}, \Sigma, q_0^\mathcal{A}, \delta^\mathcal{A}, \mathrm{col}^\mathcal{A})$ and a parity NPDTA $\mathcal{B} = (Q^\mathcal{B}, \Sigma, \Gamma, q_0^\mathcal{B}, \delta^\mathcal{B}, \perp, \mathrm{col}^\mathcal{B})$, both running over Σ-labelled X-trees for some set X, we define the product of $\mathcal{A}$ and $\mathcal{B}$ to be the following nondeterministic Muller pushdown tree automaton $\mathcal{A} \times \mathcal{B} = (Q, \Sigma, \Gamma, q_0, \delta, \perp, \mathcal{F})$:

- $Q = Q^\mathcal{A} \times Q^\mathcal{B}$
- $q_0 = (q_0^\mathcal{A}, q_0^\mathcal{B})$
- $\mathcal{F}$ consists of those subsets F of Q such that $\min\{\mathrm{col}^\mathcal{A}(\mathrm{Pr}_{Q^\mathcal{A}}(x)) \mid x \in F\}$ is even and $\min\{\mathrm{col}^\mathcal{B}(\mathrm{Pr}_{Q^\mathcal{B}}(x)) \mid x \in F\}$ is even.
- for all $(p, q) \in Q$, all $a \in \Sigma$ and all $A \in \Gamma$ such that $\delta^\mathcal{B}(q, a, A)$ is not an ϵ-transition,

$$\delta((p, q), a, A) = \bigvee_{[\phi^\mathcal{A} \in \delta^\mathcal{A}(p, a)]} \bigvee_{[\phi^\mathcal{B} \in \delta^\mathcal{B}(q, a, A)]} \bigwedge_{[x \in X]} (\downarrow_x, (p_x, q_x), \gamma_x)$$

  where $(\downarrow_x, p_x) \in \phi^\mathcal{A}$ and $(\downarrow_x, q_x, \gamma_x) \in \phi^\mathcal{B}$
- for all $(p, q) \in Q$, all $a \in \Sigma$ and all $A \in \Gamma$ such that $\delta^\mathcal{B}(q, a, A) = (\circlearrowleft_\epsilon, q', \gamma)$,

$$\delta((p, q), a, A) = (\circlearrowleft_\epsilon, (p, q'), \gamma)$$

Proposition 1. $L(\mathcal{A} \times \mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{B})$.

3 Global Specifications

Theorem 2. *The realizability problem for global deterministic contextfree specifications is undecidable for an architecture $\mathfrak{A}$ if, and only if, $C_{p_0} \neq \emptyset$ and additionally $|P_{sys}| \geq 2$ or $H_{p_0} \neq \emptyset$.*

Proof. If $C_{p_0} = \emptyset$ then the realizability problem for $\mathfrak{A}$ is just the nonemptiness problem for deterministic pushdown automata which is decidable. If $|P_{\text{sys}}| \leq 1$ and $H_{p_0} = \emptyset$ then the realizability problem for $\mathfrak{A}$ is just the usual (nondistributed) synthesis problem for deterministic contextfree winning condition which is again decidable.

Now assume that $C_{p_0} \neq \emptyset$ and $|P_{\text{sys}}| \geq 2$. To show undecidability we proceed by a reduction from the Post's Correspondence Problem PCP. Let p_1 and p_2 be system processes and let $c_{in} \in C_{p_0}$. W.l.o.g., assume that $r(c_{in}) = p_1$. Moreover, let $c_{out} \in C_{p_2}$. We silence all other channels by defining $\Sigma_c = \{\sharp\}$ for all $c \in C \setminus \{c_{in}, c_{out}\}$. Now given an instance $\mathcal{I} = ((u_0, v_0), \ldots, (u_{m-1}, v_{m-1}))$ of the PCP over an alphabet Θ (i.e., $u_i, v_i \in \Theta^*$), consider the following specification language L over $\Sigma = \Sigma_{c_{\text{in}}} \times \Sigma_{c_{\text{out}}} \times \Sigma'$ where $\Sigma_{c_{\text{in}}} = \{U, V\}$, $\Sigma_{c_{\text{out}}} = [m] \cup \Theta \cup \{\sharp\}$ and Σ' is the product of all the other (unary) alphabets which is irrelevant for the specifcation. Let L consist of all words $\alpha_{\text{in}} \frown \alpha_{\text{out}} \frown \alpha'$ where $\alpha_{\text{in}} \in \{U, V\}^\omega$ and $\alpha_{\text{out}} = \sharp i_k \ldots i_1 \sharp w \sharp \Sigma_{c_{\text{out}}}^\omega$ with $i_j \in [m]$ and $w \in \Theta^*$ such that $w = u_{i_1} \ldots u_{i_k}$ if $\alpha_{\text{in}}(0) = U$ and $w = v_{i_1} \ldots v_{i_k}$ if $\alpha_{\text{in}}(0) = V$.

Clearly, the specification language can be recognized by a deterministic parity pushdown automaton $\mathcal{P}$: Depending on the first symbol of α_{in}, $\mathcal{P}$ goes into recognition mode U or V and then, reading i_j, it pushes $u_{i_j}^R$ (being in the U-mode) or $v_{i_j}^R$ (being in the V-mode) to the stack. After the second $\sharp$, $\mathcal{P}$ checks whether the stack content coincides with the sequence w.

Notice that $|\Sigma_{\text{in}}^{p_2}| = 1$, so any strategy $\sigma_{p_2} : (\Sigma_{\text{in}}^{p_2})^* \to \Sigma_{c_{\text{out}}}$ for process p_2 uniquely determines a joint strategy for all the system processes. Moreover, $\mathcal{I}$ has a solution if, and only if, process p_2 has such a strategy σ_{p_2} which is winning: Clearly, a solution for $\mathcal{I}$ gives a winning strategy. On the other hand, since process p_2 does not observe α_{in} it just writes its output $\sharp i_k \ldots i_1 \sharp w \sharp \ldots$ independently from the recognition mode of $\mathcal{P}$. So it can only win if there is a sequence $i_k \ldots i_1$ such that $w = u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$ which implies that $\mathcal{I}$ has a solution.

To see that undecidability also holds for the case where $|P_{\text{sys}}| = 1$ but $H_{p_0} \neq \emptyset$ just notice that in the above prove, the system process p_1 is used only to block the information which is sent by the environment so that it cannot be read by process p_2. So, if there is some channel $c \in H_{p_0}$ which cannot be read by any system process, the above proof can obviously be carried out for architectures with only one system process. □

Remark 3. By a reduction from the halting problem for 2-register machines one can show that Theorem 2 also holds for deterministic 1-counter specifications.

4 Local Specifications

From now on, we consider architectures with specifications given by collections of local specifications, one for each system process. For the class of acylic architectures, it has been shown in [11], that the realizability problem for local regular specifications is decidable if, and only if, each conntected subarchitecture is a subarchitecture of a two-flanked pipeline. We continue the investigation by classifying the decidable architectures for the more general case where cycles are allowed and the local specifications may also be deterministic contextfree. In this section, we first prove decidability and undecidability results for some special classes of architectures.

4.1 Decidability

Pipelines with Backward-Channels. Let $\mathfrak{A} = (P, C, r)$ be a pipeline with backward-channels and let $C = C_f \cup C_b$ where $C_f = \bigcup_{i=1}^{n-1}\{c \in C_i | r(c) = p_{i+1}\}$ is the set of forward-channels of $\mathfrak{A}$ and $C_b = \bigcup_{i=1}^{n}\{c \in C_i | r(c) = p_j \text{ for } j \leq i\}$ is the set of backward-channels and external output-channels of $\mathfrak{A}$ and let $(\Sigma_c)_{c\in C}$ be a set of signal alphabets for the channels of $\mathfrak{A}$. Moreover, let $L_{p_1}, \ldots, L_{p_n}$ be local specifications for the system processes $p_1, \ldots, p_n$ where $L_{p_1}, \ldots, L_{p_{n-1}}$ are regular and L_{p_n} is regular or deterministic contextfree.

For any process p_i with $i \geq 1$ we define the accumulated output alphabet $\Sigma_{\text{out}}^{\geq i} := \prod_{j\geq i} \Sigma_{\text{out}}^{p_j}$ which labels all the output channels of all processes p_j with $j \geq i$ and the alphabet $\Sigma_{\text{out}}^{b,p_i} := \prod_{c\in C_b \cap C_{p_i}} \Sigma_c$ which labels all the backward-channels and external output channels of process p_i. Moreover, for $0 \leq i < n$, $\Sigma_i := \prod_{c\in C_f \cap C_{p_i}} \Sigma_c$ denotes the alphabet on the channels from p_i to p_{i+1}.

To prove decidability, we adopt the $\mathbb{B}$-labeled trees used in [11] to represent communication languages of the processes, that means, sets of infinite sequences of signals which can be sent along certain channels in the given architecture. Given an alphabet Σ, a $\mathbb{B}$-labeled Σ-tree t represents the ω-language $L^\omega(t) = \{\alpha \in \Sigma^\omega \,|\, t(\alpha{\restriction}_k) = \top \text{ for all } k \in \mathbb{N}\}$. and the language $L^*(t) = \{u \in \Sigma^* \,|\, t(u) = \top\}$ of finite words.

Now if such a tree t represents in fact a communication language, then the $\top$-labelled nodes of t form a nonempty subtree of t, containing the root of t. More formally, t has the following properties: (C1) $t(\epsilon) = \top$, (C2) if $t(u) = \bot$, then $t(ua) = \bot$ for all $a \in \Sigma$ and (C3) if $t(u) = \top$, then $t(ua) = \top$ for some $a \in \Sigma$. We call $\mathbb{B}$-labeled Σ-trees which have the properties (C1) - (C3) communication trees over Σ and we denote the set of all such trees by $\mathbb{T}_C(\Sigma)$. Notice that for $t \in \mathbb{T}_C(\Sigma)$, $L^*(t) = \{u \in \Sigma^* \,|\, u \sqsubseteq \alpha \text{ for some } \alpha \in L^\omega(t)\}$, that means, $L^*(t)$ is precisely the set of all finite prefixes of elements from $L^\omega(t)$.

Now, given a tree t_{in} which represents input sequences that a process receives and a tree t_{out} which represents output sequences that the process may write, we define the strategy product $t_{\text{in}} \hookrightarrow t_{\text{out}}$ of t_{in} and t_{out} as a set of trees t, each of which defines an assignment of input sequences from $L^\omega(t_{\text{in}})$ to output sequences from $L^\omega(t_{\text{out}})$, so it yields a strategy $\sigma(t)$ for the process.

Formally, for a tree $t_{\text{in}} \in \mathbb{T}_C(\Sigma)$ and a tree $t_{\text{out}} \in \mathbb{T}_C(\Sigma')$, the strategy product $t_{\text{in}} \hookrightarrow t_{\text{out}}$ is defined as the set of all $\mathbb{B}$-labeled $\Sigma \times \Sigma'$-trees t such that: (S1) if $t_{\text{in}}(u) = \bot$ or $t_{\text{out}}(v) = \bot$ then $t(u^\frown v) = \bot$, (S2) if $t_{\text{in}}(u) = \top$ then there is exactly one $v \in (\Sigma')^{|u|}$ such that $t(u^\frown v) = \top$ and (S3) if $t(u^\frown v) = \top$ then there is some $b \in \Sigma'$ such that for all $a \in \Sigma$ with $t_{\text{in}}(ua) = \top$ we have $t(ua^\frown vb) = \top$ and $t(ua^\frown vc) = \bot$ for $c \in \Sigma' \setminus \{b\}$. Given a tree $t_{\text{in}} \in \mathbb{T}_C(\Sigma)$, a tree $t_{\text{out}} \in \mathbb{T}_C(\Sigma')$ and a tree $t \in t_{\text{in}} \hookrightarrow t_{\text{out}}$, the strategy $\sigma(t)$ represented by t is defined as follows. For $u \in \Sigma^*$ with $t_{\text{in}}(u) = \top$, let v be the unique element from $(\Sigma')^*$ with $t(u, v) = \top$ and let a be the unique element from Σ' with $t(ub, va) = \top$ for any $b \in \Sigma$ with $t_{\text{in}}(ub) = \top$. Then, $\sigma(t)(u) = a$. Moreover, if $t_{\text{in}}(u) = \bot$ then $\sigma(t)(u) = a$ for some $a \in \Sigma'$.

Notice that in fact for any such t, $\Pr_\Sigma(L^\omega(t)) = L^\omega(t_{\text{in}})$ and $\Pr_{\Sigma'}(L^\omega(t)) \subseteq L^\omega(t_{\text{out}})$. Moreover, notice that $t_{\text{in}} \hookrightarrow t_{\text{out}} \subseteq \mathbb{T}_C(\Sigma \times \Sigma')$.

The key argument for the decidability result for pipelines with backward-channels is that a system process p_i is better informed than any system process p_j with $j > i$. In particular, a strategy for p_i needs only to depend on the input that it receives from the previous process p_{i-1} and not on the input received via backward-channels.

Lemma 4. *There is a joint winning strategy $\sigma = (\sigma_1, \ldots, \sigma_n)$ for the system processes if, and only if, there are functions $\tau_i \colon \Sigma_{i-1}^* \to \Sigma_{out}^{p_i}$ for $i = 1, \ldots, n$ such that any global system behavior which is consistent with $\tau = (\tau_1, \ldots, \tau_n)$ is in the global system specification.*

Proof. If there are such functions τ_i, then we define σ_i for $i = 1, \ldots, n$ by $\sigma_i(u) = \tau_i(\Pr_{\Sigma_{i-1}})(u))$. Obviously, any global system behavior which is consistent with σ is also consistent with τ and is therefore in the global system specification. Hence, σ is a joint winning strategy for the system processes.

Now let conversely σ be a joint winning strategy for the system processes. First we show that, for any $1 \leq i \leq n$ and any $u \in \Sigma_{i-1}^*$, there is exactly one $v \in (\Sigma_{\text{out}}^{\geq i})^{|u|}$ such that $u^\frown v$ is consistent with $(\sigma_i, \ldots, \sigma_n)$. We show this, for any i, by induction on $|u|$. As $u = \epsilon$ is trivial, let $|u| > 0$ and let $u = u'a$ for some $a \in \Sigma_{i-1}$. By induction hypothesis, there is a $v' \in (\Sigma_{\text{out}}^{\geq i})^{|u'|}$ such that $u'^\frown v'$ is consistent with $(\sigma_i, \ldots, \sigma_n)$ and we define $v_j := \sigma_j(\Pr_{\Sigma_{\text{in}}^{p_j}}(v'))$ for $j \geq i$. Obviously, $\hat{v} = v_i^\frown \ldots^\frown v_n \in \Sigma_{\text{out}}^{\geq i})^{|u|}$ and $u^\frown \hat{v}$ is consistent with $(\sigma_i, \ldots, \sigma_n)$. Now let $v \in (\Sigma_{\text{out}}^{\geq i})^{|u|}$ such that $u^\frown v$ is consistent with $(\sigma_i, \ldots, \sigma_n)$. We show that $v = \hat{v}$. Notice that $u'^\frown(v^{-1})$ is consistent with $(\sigma_i, \ldots, \sigma_n)$ so, by induction hypothesis, $v^{-1} = \hat{v}^{-1}$. Moreover, as $u^\frown v$ is consistent with $(\sigma_i, \ldots, \sigma_n)$, we have $\Pr_{\Sigma_{\text{out}}^{p_j}}(v) = \sigma_j(\Pr_{\Sigma_{\text{in}}^{p_j}}(v^{-1})) = \sigma_j(\Pr_{\Sigma_{\text{in}}^{p_j}}(\hat{v}^{-1})) = \Pr_{\Sigma_{\text{out}}^{p_j}}(\hat{v})$ and as $\Sigma_{\text{out}}^{\geq i} = \prod_{j=i}^n \Sigma_{\text{out}}^{p_j}$ we have $v = \hat{v}$.

Now we define the functions τ_i for $i = 1, \ldots, n$ as follows. For $u \in \Sigma_{i-1}^*$, let $v \in (\Sigma_{\text{out}}^{\geq i})^{|u|}$ be the unique word such that $u^\frown v$ is consistent with $(\sigma_i, \ldots, \sigma_n)$ and let $\tau_i(u) = \sigma_i(u^\frown \Pr_{\Sigma_{\text{in}}^{p_i}}(v))$. To prove that τ is winning for the system processes, consider any global system behavior α which is consistent with τ. By induction on k we show that any finite prefix $\alpha{\uparrow_k}$ of α is consistent with σ.

Clearly, $\alpha\!\upharpoonright_0$ is consistent with σ, so let $k > 0$ and let $1 \leq i \leq n$. As $\alpha\!\upharpoonright_{k-1}$ is consistent with σ, $\mathrm{Pr}_{\Sigma_{i-1}}(\alpha\!\upharpoonright_{k-1}) \frown \mathrm{Pr}_{\Sigma_{\mathrm{out}}^{\geq i}}(\alpha\!\upharpoonright_{k-1})$ is consistent with $(\sigma_i, \ldots, \sigma_n)$. Moreover, since α is consistent with τ, the definition of τ_i yields $\mathrm{Pr}_{\Sigma_{\mathrm{out}}^{p_i}}(\alpha\!\upharpoonright_k) = \tau_i(\mathrm{Pr}_{\Sigma_{i-1}}(\alpha\!\upharpoonright_{k-1})) = \sigma_i(\mathrm{Pr}_{\Sigma_{\mathrm{in}}^{p_i}}(\alpha\!\upharpoonright_{k-1}))$ so $\alpha\!\upharpoonright_k$ is consistent with σ_i. As i has been chosen arbitrarily, $\alpha\!\upharpoonright_k$ is consistent with σ. Therefore, α is consistent with σ and hence, α is in the global system specification. □

Due to this observation, the following definition of an extended local strategy is meaningful. For $1 \leq i \leq n$, an extended local strategy for process p_i is a tuple $\sigma_{\geq i} = (\sigma_i, \ldots, \sigma_n)$ of functions $\sigma_j \colon (\Sigma_{i-1})^* \to \Sigma_{\mathrm{out}}^{p_j}$, i.e., it takes the local input history of process p_i, ignoring the backward-channels read by p_i, and yields the next output symbol for each process p_j with $j \geq i$. Such a strategy is called locally winning on inputs from $L_{\mathrm{in}} \subseteq \Sigma_{i-1}^{\omega}$, if each global system behavior α of $\mathfrak{A}$ with $\mathrm{Pr}_{\Sigma_{i-1}}(\alpha) \in L_{\mathrm{in}}$ which is consistent with $\sigma_{\geq i}$ fulfills L_{p_i}, i.e., $\mathrm{Pr}_{\Sigma^{p_i}}(\alpha) \in L_{p_i}$. The main technical argument how these strategies can be used to decide the realizability problem for $\mathfrak{A}$ is given in the following Lemma. We give here only a rough proof sketch, a full proof can be found in Appendix A.

Lemma 5. *For any $1 \leq i < n$ there is a parity NTA $\mathcal{N}_i$ over $\mathbb{B}$-labeled $\Sigma_{out}^{\geq i}$-trees which accepts a tree $t_{out} \in \mathbb{T}_C(\Sigma_{out}^{\geq i})$ if, and only if, there are a $\mathbb{B}$-labeled Σ_{i-1}-tree $t_{in} \in \mathbb{T}_C(\Sigma_{i-1})$, a tree $t \in t_{in} \hookrightarrow t_{out}$ and strategies $\sigma_1, \ldots, \sigma_{i-1}$ for processes $p_1, \ldots, p_{i-1}$ such that $\sigma(t)$ is locally winning on $L^{\omega}(t_{in})$ and*

- *$\sigma_1 \circ Pr_{\Sigma_1} \circ \ldots \circ \sigma_{i-1} \circ Pr_{\Sigma_{i-1}}$ generates a language $L \subseteq L^{\omega}(t_{in})$ over Σ_0^{ω}*
- *$(\sigma_1, \ldots, \sigma_{i-1}, \sigma(t))$ is winning for $p_1, \ldots, p_i$.*

Proof. (Sketch) We prove this by induction on i. We omit the base case $i = 1$ and consider only the case $i > 1$. For this, let $\mathcal{N}_{i-1}$ be a parity NTA over $\mathbb{B}$-labeled $\Sigma_{\mathrm{out}}^{\geq i-1}$-trees according to the induction hypothesis. Then there is a parity NTA $\mathcal{N}'_{i-1}$ over $\mathbb{B}$-labeled $\Sigma_{i-1} \times \Sigma_{\mathrm{out}}^{\geq i}$-trees which accepts a tree t if, and only if, $\mathrm{wide}(t, \Sigma_{\mathrm{out}}^{b, p_{i-1}}) \in L(\mathcal{N}_{i-1})$. Now we construct a parity ATA $\mathcal{A}_i$ over $\mathbb{B}$-labeled $\Sigma_{\mathrm{out}}^{\geq i}$-trees which, roughly, works as follows: Running on a tree t_{out}, in each step, $\mathcal{A}_i$ guesses an output signal $b \in \Sigma_{\mathrm{out}}^{\geq i}$ and a set $\emptyset \neq X \subseteq \Sigma_{i-1}$ of possible input signals and sends, for each $(x, y) \in \Sigma_{i-1} \times \Sigma_{\mathrm{out}}^{\geq i}$, a copy into direction y. If $x \in X$ and $y = b$, it sends a $\top$-copy, otherwise it sends a $\bot$-copy. Moreover, if $\mathcal{A}_i$ encounters a $\bot$-symbol in t_{out} when being in a $\top$-copy (that means, output b should not have been chosen in the previous step according to t_{out}), it immediately rejects. In this way, $\mathcal{A}_i$ guesses a tree $t_{\mathrm{in}} \in \mathbb{T}_C(\Sigma_{i-1})$ and a tree $t \in t_{\mathrm{in}} \hookrightarrow t_{\mathrm{out}}$. While doing so, $\mathcal{A}_i$ simulates $\mathcal{N}'_{i-1}$ on t and it simulates a deterministic parity automaton recognizing L_{p_i} on all paths $\alpha \in L^{\omega}(t)$. Therefore, $\mathcal{A}_i$ accepts iff $t \in L(\mathcal{N}'_{i-1})$ and $\sigma(t)$ is locally winning on $L^{\omega}(t_{\mathrm{in}})$. Now using the induction hypothesis, one can show that $\mathcal{A}_i$ fulfills all conditions of the lemma. Moreover, there is a parity NTA $\mathcal{N}_i$ with $L(\mathcal{N}_i) = L(\mathcal{A}_i)$, which concludes the proof. □

For the last process p_n, we need the following Lemma.

Lemma 6. *There is a parity NPDTA $\mathcal{N}_n$ over $\mathbb{B}$-labeled $\Sigma_{n-1} \times \Sigma_{out}^{p_n}$-trees which accepts a tree t if, and only if, $t \in t_{in} \hookrightarrow t_{out}$ for some $t_{in} \in \mathbb{T}_C(\Sigma_{n-1})$ and some $t_{out} \in \mathbb{T}_C(\Sigma_{out}^{p_n})$ such that $\sigma(t)$ is locally winning on $L^\omega(t_{in})$.*

Proof. We abbreviate $\Sigma = \Sigma_{n-1}$ and $\Sigma' = \Sigma_{\mathrm{out}}^{p_n}$ and we consider a deterministic PDA $\mathcal{S} = (Q^{\mathcal{S}}, \Sigma \times \Sigma', \Gamma^{\mathcal{S}}, \delta^{\mathcal{S}}, q_0^{\mathcal{S}}, \bot, \mathrm{col}^{\mathcal{S}})$ with $L(\mathcal{S}) = L_{p_n}$. We define $\mathcal{A} = (Q, \mathbb{B}, \Gamma, \delta, q_0, \bot, \mathrm{col})$ as follows.

- $Q = Q^{\mathcal{S}} \cup \{q_{reject}, q_\bot\}$
- $\Gamma = \Gamma^{\mathcal{S}}$
- $\mathrm{col}(q) = \mathrm{col}^{\mathcal{S}}(q)$ for all $q \in Q^{\mathcal{S}}$
- $\mathrm{col}(q_{reject}) = 1$ and $\mathrm{col}(q_\bot) = 0$
- $q_0 = q_0^{\mathcal{S}}$
- for $q \in Q^{\mathcal{S}}$ and $A \in \Gamma$ with $\delta^{\mathcal{S}}(q, \epsilon, A) \neq \emptyset$,

$$\delta(q, \top, A) = (\circlearrowleft_\epsilon, \delta^{\mathcal{S}}(q, \epsilon, A))$$

- for $q \in Q^{\mathcal{S}}$ and $A \in \Gamma$ with $\delta^{\mathcal{S}}(q, \epsilon, A) = \emptyset$,

$$\delta(q, \top, A) = \bigvee_{[\emptyset \neq X \subseteq \Sigma]} \bigvee_{[a \in \Sigma']} \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_{(x,y)}, (q_{xy}, \gamma_{xy}))$$

  where $(q_{xy}, \gamma_{xy}) = \begin{cases} \delta^{\mathcal{S}}(q, (x,y), A) & \text{if } (x,y) \in X \times \{a\} \\ (q_\bot, A) & \text{else} \end{cases}$

- for $A \in \Gamma$, $\delta(q_\bot, \bot, A) = \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_{(x,y)}, q_\bot, A)$
- for $A \in \Gamma$, $\delta(q_\bot, \top, A) = \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_{(x,y)}, q_{reject}, A)$
- for $q \in Q^{\mathcal{S}}$ and $A \in \Gamma$,
  $\delta((q, \top), \bot, A) = \delta((q, \bot), \top, A) = \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_{(x,y)}, q_{reject}, A)$
- for $(\zeta, \zeta') \in \mathbb{B}^2$ and $A \in \Gamma$,
  $\delta((q_{reject}, \zeta), \zeta') = \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_{(x,y)}, q_{reject}, A)$

For the correctness of the automaton, we give here only the idea: $\mathcal{A}$ guesses a strategy $\sigma\colon \Sigma^* \to \Sigma'$ and checks, whether this strategy is represented by t, i.e., $\sigma(t) = \sigma$. For this, being in a node $u^\frown v \in (\Sigma \times \Sigma')^*$, $\mathcal{A}$ guesses the answer a of σ to the input sequence u and it also guesses a subset $X \subseteq \Sigma'$ of possible input signals which it may receive in the next step. Then the elements from $X \times \{a\}$ are exactly the signals which may occur in the next step, so precisely those successors of $u^\frown v$ should be labeled $\top$ and the automaton checks this, by sending a corresponding copy of the specification automaton $\mathcal{S}$ to those successors and a distinguished state $q_\bot$ to all other successors. □

Now by applying a widening argument for $\Sigma_{\mathrm{out}}^{b,p_{n-1}}$, similar as in the proof of Lemma 5, one obtains an automaton $\mathcal{N}'_{n-1}$ from $\mathcal{N}_{n-1}$ and there is a parity NPDTA $\mathcal{N}$ recognizing the intersection of $L(\mathcal{N}'_{n-1})$ and $L(\mathcal{N}_n)$, where $\mathcal{N}_n$ is the parity NPDTA obtained from Lemma 6. Then one can show that $(L_{p_1}, \ldots, L_{p_n})$ is realizable in $\mathfrak{A}$ if, and only if, $L(\mathcal{N}) \neq \emptyset$ and as nonemptiness of $L(\mathcal{N})$ is decidable, the decidability result is established.

Theorem 7. *The realizability problem for $\mathfrak{A}$ is decidable if $L_{p_1}, \ldots, L_{p_{n-1}}$ are regular and L_{p_n} is regular or deterministic contextfree.*

Proof. First, if $n = 1$, the theorem follows easily from Lemma 6. So let $n > 1$ and let $\mathcal{N}_{n-1}$ be the parity NTA over $\mathbb{B}$-labeled $\Sigma_{\text{out}}^{\geq n-1}$-trees according to Lemma 5. Then there is a parity ATA $\mathcal{A}'_{n-1}$ over $\mathbb{B}$-labeled $\Sigma_{n-1} \times \Sigma_{\text{out}}^{\geq n}$-trees which accepts a tree t if, and only if, $\text{wide}(t, \Sigma_{\text{out}}^{b,p_{n-1}}) \in L(\mathcal{N}_{n-1})$. Furthermore, $\mathcal{A}'_{n-1}$ can be transformed into an equivalent parity NTA $\mathcal{N}'_{n-1}$. Moreover, let $\mathcal{N}_n$ be the parity NPDTA over $\mathbb{B}$-labeled $\Sigma_{n-1} \times \Sigma_{\text{out}}^{p_n}$-trees according to Lemma 6. Then there is a parity NPDTA $\mathcal{N}$ over $\mathbb{B}$-labeled $\Sigma_{n-1} \times \Sigma_{\text{out}}^{p_n}$-trees such that $L(\mathcal{N}) = L(\mathcal{N}'_{n-1}) \cap L(\mathcal{N}_n) \neq \emptyset$. Now we prove that $L_{p_1}, \ldots, L_{p_n}$ are realizable in $\mathfrak{A}$ if $L(\mathcal{N}) \neq \emptyset$. The converse direction can be proved using similar ideas. As emptiness of $L(\mathcal{N})$ is decidable, the theorem follows.

So, let $t \in L(\mathcal{N})$. As $t \in L(\mathcal{N}_n)$, $t \in t_{\text{in}} \hookrightarrow t_{\text{out}}$ for some $t_{\text{in}} \in \mathbb{T}_C(\Sigma_{n-1})$ and some $t_{\text{out}} \in \mathbb{T}_C(\Sigma_{\text{out}}^{p_n})$ such that $\sigma(t)$ is locally winning on $L^\omega(t_{\text{in}})$. Moreover, as $t \in L(\mathcal{N}'_{n-1})$, $s_{\text{out}} = \text{wide}(t, \Sigma_{\text{out}}^{b,p_{n-1}}) \in L(\mathcal{N}_{n-1})$ that means, there are a $\mathbb{B}$-labeled Σ_{n-2}-tree s_{in}, a tree $s \in s_{\text{in}} \hookrightarrow s_{\text{out}}$ and strategies $\sigma_1, \ldots, \sigma_{n-2}$ for processes $p_1, \ldots, p_{n-2}$ such that $\sigma_1 \circ \text{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{n-2} \circ \text{Pr}_{\Sigma_{n-2}}$ generates a language $L \subseteq L^\omega(s_{\text{in}})$ over Σ_0^ω and $(\sigma_1, \ldots, \sigma_{n-2}, \sigma(s))$ is winning for $p_1, \ldots, p_{n-1}$.

We denote $\sigma(s) = (\sigma_{n-1}, \tau_n)$ and $\sigma(t) = \sigma_n$. It is easy to show that the language generated by $\sigma_1 \circ \text{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{n-2} \circ \text{Pr}_{\Sigma_{n-2}} \circ \sigma_{n-1} \circ \text{Pr}_{\Sigma_{n-1}}$ over Σ_0^ω is a subset of $L^\omega(t_{\text{in}}) = L^\omega_{\Sigma_{n-1}}(t)$.

Now we show that $\sigma_n(\epsilon) = \tau_n(\epsilon)$ and $\sigma_n(\text{Pr}_{\Sigma_{n-1}}(\sigma^*_{n-1}(u^{-1}))) = \tau_n(u)$ for all $u \in L^*(s_{\text{in}}) = L^*_{\Sigma_{n-2}}(s) \subseteq \Sigma^*_{n-2}$ with $|u| \geq 1$. First, by definition of $\sigma(s)$, for all $a \in \Sigma_{n-2}$ with $s_{\text{in}}(a) = \top$ we have $s((a, \sigma(s)(\epsilon))) = \top$ which, by definition of $s_{\text{in}} \hookrightarrow s_{\text{out}}$, implies $s_{\text{out}}(\sigma(s)(\epsilon)) = \top$. Since $s_{\text{out}} = \text{wide}(t, \Sigma_{\text{out}}^{b,p_{n-1}})$ this yields $t(\text{Pr}_{\Sigma_{n-1} \times \Sigma_{\text{out}}^{\geq n}}(\sigma(s)(\epsilon))) = \top$ so, by definition of $\sigma(t)$, $\sigma_n(\epsilon) = \sigma(t)(\epsilon) = \text{Pr}_{\Sigma_{\text{out}}^{\geq n}}(\sigma(s)(\epsilon)) = \tau_n(\epsilon)$. Now let $u \in L^*_{\Sigma_{n-2}}(s)$ with $|u| \geq 1$, that means, there is some $v \in (\Sigma_{\text{out}}^{\geq n-1})^{|u|}$ with $s(u^\frown v) = \top$. By definition of $\sigma(s)$ we have $v = \sigma(s)^*(u^{-1})$ and, for all $a \in \Sigma_{n-2}$ with $s_{\text{in}}(ua) = \top$, $s(ua^\frown v\sigma(s)(u)) = \top$ which, by definition of $s_{\text{in}} \hookrightarrow s_{\text{out}}$, implies $s_{\text{out}}(v\sigma(s)(u)) = \top$. This yields $t(\text{Pr}_{\Sigma_{n-1} \times \Sigma_{\text{out}}^{\geq n}}(v\sigma(s)(u))) = \top$ and as $\text{Pr}_{\Sigma_{n-1}}(\sigma^*_{n-1}(u^{-1})) = \text{Pr}_{\Sigma_{n-1}}(\sigma(s)^*(u^{-1}))$, by definition of $\sigma(t)$, $\sigma_n(\text{Pr}_{\Sigma_{n-1}}(\sigma^*_{n-1}(u^{-1}))) = \sigma(t)(\text{Pr}_{\Sigma_{n-1}}(\sigma(s)^*(u^{-1}))) = \sigma(t)(\text{Pr}_{\Sigma_{n-1}}(v)) = \text{Pr}_{\Sigma_{\text{out}}^{\geq n}}(\sigma(s)(u)) = \tau_n(u)$.

It remains to prove that $(\sigma_1, \ldots, \sigma_{n-2}, \sigma_{n-1}, \sigma(t))$ is winning for $p_1, \ldots, p_n$. By Lemma 5, $(\sigma_1, \ldots, \sigma_{n-2}, \sigma(s))$ is winning for $p_1, \ldots, p_{n-1}$. Moreover, the language generated by $\sigma_1 \circ \text{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{n-2} \circ \text{Pr}_{\Sigma_{n-2}} \circ \sigma_{n-1} \circ \text{Pr}_{\Sigma_{n-1}}$ over Σ_0^ω is a subset of $L^\omega(t_{\text{in}})$ and σ_n is locally winning on $L^\omega(t_{\text{in}})$. Furthermore, as we have shown above, $\tau_n(\epsilon) = \sigma_n(\epsilon)$ and $\tau_n(u) = \sigma_n(\text{Pr}_{\Sigma_{n-1}}(\sigma^*_{n-1}(u^{-1})))$ for all $u \in L^*(s_{\text{in}})$ with $|u| \geq 1$. Now, if α is some global system behavior which is consistent with $(\sigma_1, \ldots, \sigma_{n-2}, \sigma_{n-1}, \sigma_n)$ then this shows that α is also consistent with $(\sigma_1, \ldots, \sigma_{n-2}, \sigma_{n-1}, \tau_n)$, so all specifications $L_{p_1}, \ldots, L_{p_{n-1}}$ are satisfied and as σ_n is locally winning on $L^\omega(t_{\text{in}})$, L_{p_n} is satisfied as well. □

Two-Flanked Pipelines with Backward-Channels. Let $\mathfrak{A} = (P, C, r)$ be a two-flanked pipeline with backward-channels, let $(\Sigma_c)_{c \in C}$ be a labeling of $\mathfrak{A}$ and let $L_{p_1}, \ldots, L_{p_n}$ be regular local specifications for the system processes.

First we consider the case where there are no backward-channels from the last process. The main idea and the constructions are essentially the same as for the case of pipelines with backward-channels. Clearly, the construction has to be adapted to account for the fact, that processes p_i for $i < n$ can determine the decisions of all processes p_j with $i \leq j < n$, but not those of process p_n, as a strategy for process p_n depends not only on the input from p_{n-1} but also on a part of the input from the environment. However, since these decisions are not relevant for the satisfaction of the local specification of p_i, it is not necessary that process p_i makes those decision.

Lemma 4 and the definition of an extended local strategy have to be adapted as follows.

Lemma 8. *There is a joint winning strategy $\sigma = (\sigma_1, \ldots, \sigma_n)$ for the system processes if, and only if, there are functions $\tau_i : \Sigma_{i-1}^* \to \Sigma_{out}^{p_i}$ for $i = 1, \ldots, n-1$ and a local strategy τ_n for process p_n such that any global system behavior which is consistent with $(\tau_1, \ldots, \tau_n)$ is in the global system specification.*

So, an extended local strategy for p_i is now a tuple $\sigma_{\geq i} = (\sigma_i, \ldots, \sigma_{n-1})$ of functions $\sigma_j \colon (\Sigma_{i-1})^* \to \Sigma_{\text{out}}^{p_j}$.

Such a strategy is called locally winning on a language $L_{\text{in}} \subseteq (\Sigma_{i-1})^\omega$ if any global system behavior α with $\Pr_{\Sigma_{i-1}}(\alpha) \in L_{\text{in}}$ which is consistent with $(\sigma_i, \ldots, \sigma_{n-1})$ fulfills $\Pr_{\Sigma^{p_i}}(\alpha) \in L_{p_i}$. Moreover, if $\sigma_{\geq i} = (\sigma_i, \ldots, \sigma_{n-1})$ is an extended local strategy for process p_i and $\sigma_1, \ldots, \sigma_{i-1}$ are strategies for processes $1, \ldots, i-1$, then $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_{\geq i})$ is called winning for processes $p_1, \ldots, p_i$, if any global system behavior α which is consistent with $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_{\geq i})$ fulfills $\Pr_{\Sigma^{p_j}}(\alpha) \in L_{p_j}$ for $j = 1, \ldots, i$.

The accumulated output alphabets are now defined by $\Sigma_{\text{out}}^{\geq i} := \prod_{j=i}^{n-1} \Sigma_{\text{out}}^{p_j}$. Moreover, due to the additional input channel from the environment, we define $\Sigma_{01} := \prod_{c \in C_0, r(c) = p_1} \Sigma_c$ and $\Sigma_{0n} := \prod_{c \in C_0, r(c) = p_n} \Sigma_c$. The alphabets Σ_i for $i = 1, \ldots, n-1$ and $\Sigma_{\text{out}}^{b, p_i}$ for $i = 1, \ldots, n$ are defined as before.

Now Lemma 5 holds just as before with the new definition of $\Sigma_{\text{out}}^{\geq i}$ and the new notion of an extended local strategy. Lemma 6 however, has to be reformulated as follows.

Lemma 9. *There is a parity NPDTA $\mathcal{N}_n$ over $\mathbb{B}$-labelled Σ_{n-1}-trees which accepts a tree $t_{in} \in \mathbb{T}_C(\Sigma_{n-1})$ if, and only if, there is a local strategy for process p_n which is locally winning on $\{\alpha^\frown\beta \mid \alpha \in L^\omega(t_{in}), \beta \in \Sigma_{0n}^\omega\}$.*

Proof. We abbreviate $\Sigma := \Sigma_{0n}$ and $\Sigma' := \Sigma_{\text{out}}^{p_n}$. Let $\mathcal{S} = (Q^{\mathcal{S}}, \Sigma^{\mathcal{S}}, \delta^{\mathcal{S}}, q_0^{\mathcal{S}}, \text{col}^{\mathcal{S}})$ be a deterministic parity automaton such that $L(\mathcal{S}) = L_{p_n}$. We define the alternating parity ATA $\mathcal{A} = (Q, \mathbb{B}, \delta, q_0, \text{col})$ as follows.

- $Q = (Q^{\mathcal{S}} \cup \{q_{accept}\}) \times [\Sigma_\epsilon \times \Sigma'_\epsilon]$
- $q_0 = (q_0^{\mathcal{S}}, [\epsilon, \epsilon])$

– $\mathrm{col}(q,[a_2,a_3]) = \mathrm{col}(q)$ for all $(q,a) \in (Q^{\mathcal{S}} \times ([\Sigma_\epsilon \times \Sigma'_\epsilon])$ and $\mathrm{col}(q_{accept}) = 0$
– for $(q,[a_2,a_3]) \in Q^{\mathcal{S}} \times [\Sigma_\epsilon \times \Sigma'_\epsilon]$

$$\delta((q,[a_2,a_3]),\top) = \bigvee_{[b_3 \in \Sigma']} \bigwedge_{[(b_1,b_2) \in \Sigma_{n-1} \times \Sigma]} (\downarrow_{b_1}, (\delta^{\mathcal{S}}(q,(b_1,b_2,b_3)),[b_2,b_3]))$$

– for $(q,[a_2,a_3]) \in Q$,

$$\delta((q,[a_2,a_3]),\bot) = \bigvee_{[b_3 \in \Sigma']} \bigwedge_{[(b_1,b_2) \in \Sigma_{n-1} \times \Sigma]} (\downarrow_{b_1}, (q_{accept},[b_2,b_3]))$$

□

Now Lemma 5 holds just as before with the new definition of $\Sigma_{\mathrm{out}}^{\geq i}$ and the new notion of an extended local strategy. Lemma 6 however, has to be reformulated.

Lemma 10. *There is a parity NTA $\mathcal{N}_n$ over $\mathbb{B}$-labeled Σ_{n-1}-trees which accepts a tree $t_{in} \in \mathbb{T}_C(\Sigma_{n-1})$ if, and only if, there is a local strategy for process p_n which is locally winning on $\{\alpha \in (\Sigma_{in}^{p_n})^\omega \mid Pr_{\Sigma_{n-1}}(\alpha) \in L^\omega(t_{in})\}$.*

Theorem 11. *The realizability problem for regular local specifications is decidable for $\mathfrak{A}$ if there is no backward-channel from the last process p_n.*

Now we prove decidability for the case $n = 2$, where we also have backward-channels from the last process, so let $P = \{p_0, p_1, p_2\}$, let $\Gamma_i = \prod_{c \in C_0, r(c) = p_i} \Sigma_c$, $\Sigma_1 = \prod_{c \in C_1, r(c) = p_2} \Sigma_c$ and Σ_2 analogous. For convenience, we omit the external output channels of p_1 and p_2, it is straightforward how to account for these channels in the proof given below, if they are present.

For the proof, we extend the $\mathbb{B}$-labeled trees as here, ω-languages over $\Sigma = \Sigma_1 \times \Sigma_2$ are represented, but we want to maintain access to the components. A $\mathbb{B}^2$-labeled Σ-tree t represents the language $L^\omega(t) \subseteq \Sigma^\omega$ with $\alpha \in L^\omega(t)$ if, and only if, for each finite prefix w of α we have $t(w) = (\top, \top)$. However, more information is stored in such a tree: the components $\mathrm{Pr}_{\Sigma_i}(\alpha)$ may depend on each other which is expressed in the inidividual components of the $\mathbb{B}$-tuples. If $t(u^\frown v) = (\top, \bot)$, then this tells us that $v{\uparrow_{|v|-1}}$ may be answered by u but $u{\uparrow_{|u|-1}}$ may not be answered by v, analogous for $t(u^\frown v) = (\bot, \top)$. Clearly this is different from just saying that $u^\frown v$ will not occur.

Of course, any such tree t which in fact represents a joint output language of p_1 and p_2 has properties analogous to the properties (C1) - (C3) of the communication trees $\mathbb{T}_C(\Sigma)$. However, as we don't need those properties in the decidability proof, we do not require them explicitly.

Theorem 12. *The realizability problem for regular local specifications is decidable for any two-flanked pipeline with backward-channels and $n = 2$.*

Proof. We construct two parity ATA $\mathcal{A}_1$ and $\mathcal{A}_2$ over $\mathbb{B}^2$-labeled Σ-trees which, roughly, work as follows. When running on a tree t, at each step, $\mathcal{A}_1$ guesses an output signal $b \in \Sigma_1$ and sends, for any possible input signal $(x, y) \in \Gamma_1 \times \Sigma_2$, a

copy into direction (b, y). If the corresponding node is labeled with $\bot$ in the Σ_1-component (that means, output b should not be chosen in this situation according to t), then $\mathcal{A}_1$ rejects immediately and if it is labeled with $\bot$ in the Σ_2-component (that means, input y will not occur in the next step according to t), $\mathcal{A}_1$ goes into a special accepting state. This way, $\mathcal{A}_1$ guesses a local strategy for p_1 on all inputs from Γ_1 and those inputs from Σ_2 which it may receive according to the intended joint strategy for p_1 and p_2. Moreover, on all paths which are consistent with this strategy, it simulates a deterministic parity automaton, recognizing the local specification L_{p_1}. The automaton $\mathcal{A}_2$ works analogously and thus one can show that $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \neq \emptyset$ iff a joint winning strategy for p_1 and p_2 exists.

To be more formal, consider parity automata $\mathcal{S}_i = (Q^{\mathcal{S}_i}, \Sigma^{p_i}, \delta^{\mathcal{S}_i}, q_0^{\mathcal{S}_i}, \mathrm{col}^{\mathcal{S}_i})$ with $L(\mathcal{S}_i) = L_{p_i}$ for $i = 1, 2$. We construct the alternating parity tree automata $\mathcal{A}_i = (Q^i, \mathbb{B}^2, \delta^i, q_0^i, \mathrm{col}^i)$ $i = 1, 2$ over $\mathbb{B}^2$-labeled $\Sigma_1 \times \Sigma_2$-trees as follows. We only describe $\mathcal{A}_1 = (Q, \mathbb{B}^2, \delta, q_0, \mathrm{col})$ using $\mathcal{S}_1 = (Q^{\mathcal{S}}, \Sigma_{p_1}, \delta^{\mathcal{S}}, q_0^{\mathcal{S}}, \mathrm{col}^{\mathcal{S}})$, $\mathcal{A}_2$ is defined completely analogously.

- $Q = (Q^{\mathcal{S}} \uplus \{q_{accept}, q_{reject}\}) \times (\Gamma_1)_\epsilon$
- $q_0 = (q_0^{\mathcal{S}}, \epsilon)$
- $\mathrm{col}(q, a_1) = \mathrm{col}^{\mathcal{S}}(q)$ for $(q, a_1) \in Q^{\mathcal{S}} \times (\Gamma_1)_\epsilon$
- $\mathrm{col}((q_{accept}, a_1)) = 0$ and $\mathrm{col}((q_{reject}, a_1)) = 1$ for all $a_1 \in (\Gamma_1)_\epsilon$
- for $(q, a_1) \in Q^{\mathcal{S}} \times (\Gamma_1)_\epsilon$,

$$\delta((q, a_1), (\top, \top)) = \bigvee_{[b_3 \in \Sigma_1]} \bigwedge_{[(b_1, b_2) \in \Gamma_1 \times \Sigma_2]} (\downarrow_{(b_3, b_2)}, (\delta^{\mathcal{S}}(q, (b_1, b_2, b_3)), b_1))$$

- for $(q, a_1) \in Q^{\mathcal{S}} \times (\Gamma_1)_\epsilon$ and $\zeta_2 \in \mathbb{B}$,
  $\delta((q, a_1), (\bot, \zeta_2)) = \bigvee_{[b_3 \in \Sigma_1]} \bigwedge_{[(b_1, b_2) \in \Gamma_1 \times \Sigma_2]} (\downarrow_{(b_3, b_2)}, (q_{reject}, b_1))$
- for $a_1 \in (\Gamma_1)_\epsilon$ and $(\zeta_1, \zeta_2) \in \mathbb{B}^2$,
  $\delta((q_{reject}, a_1), (\zeta_1, \zeta_2)) = \bigvee_{[b_3 \in \Sigma_1]} \bigwedge_{[(b_1, b_2) \in \Gamma_1 \times \Sigma_2]} (\downarrow_{(b_3, b_2)}, (q_{reject}, b_1))$
- for $(q, a_1) \in Q^{\mathcal{S}} \times (\Gamma_1)_\epsilon$,
  $\delta((q, a_1), (\top, \bot)) = \bigvee_{[b_3 \in \Sigma_1]} \bigwedge_{[(b_1, b_2) \in \Gamma_1 \times \Sigma_2]} (\downarrow_{(b_3, b_2)}, (q_{accept}, b_1))$
- for $(\zeta_1, \zeta_2) \in \mathbb{B}$ and $a_1 \in (\Gamma_1)_\epsilon$,
  $\delta((q_{accept}, a_1), (\zeta_1, \zeta_2)) = \bigvee_{[b_3 \in \Sigma_1]} \bigwedge_{[(b_1, b_2) \in \Gamma_1 \times \Sigma_2]} (\downarrow_{(b_3, b_2)}, (q_{accept}, b_1))$

Now we claim that $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \neq \emptyset$ if, and only if, there is a joint winning strategy $\sigma = (\sigma_1, \sigma_2)$ for p_1 and p_2. To prove this, first let $t \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Moreover, for $i = 1, 2$, let $\rho_i : T^i \to (\Sigma_1 \times \Sigma_2)^* \times Q^i$ with $T^i \subseteq \mathbb{N}^*$ be a run of $\mathcal{A}_i$ on t. We define the strategy σ_1 according to ρ_1 as follows. Using the definition of $\mathcal{A}_1$ an easy induction on k yields that, for any $\alpha_1{\uparrow_k}^\frown\alpha_2{\uparrow_k} \in (\Gamma_1 \times \Sigma_2)^*$, the following holds. There is exactly one vertex x in ρ_1 such that the unique path from the root of ρ_1 to x is labeled with $\alpha_1{\uparrow_k}^\frown\alpha_2{\uparrow_k}$ in the $\Gamma_1 \times \Sigma_2$-components and there is some $b_3 \in \Sigma_1$ such that any successor of x in ρ_1 is labeled with b_3

in the Σ_1-component. Now we define $\sigma_1(\alpha_1{\uparrow_k}^\frown\alpha_2{\uparrow_k}) = b_3$. The strategy σ_2 is defined completely analogously, using the run ρ_2.

Now let $\alpha = \alpha_1{}^\frown\alpha_2{}^\frown\alpha_3{}^\frown\alpha_4 \in (\Gamma_1 \times \Gamma_2 \times \Sigma_1 \times \Sigma_2)^\omega$ be a global system behavior which is conistent with the joint strategy $\sigma = (\sigma_1, \sigma_2)$. First, as above, there is exactly one path π_1 in ρ_1 which is labeled with $\alpha_1{}^\frown\alpha_4$ in the $\Gamma_1 \times \Sigma_2$-components and there is exactly one path π_2 in ρ_2 which is labeled with $\alpha_2{}^\frown\alpha_3$ in the $\Gamma_2 \times \Sigma_1$-components. Now as α is consistent with σ, $\beta_1 := \alpha_1{}^\frown\alpha_4{}^\frown\alpha_3$ is consistent with σ_1 and $\beta_2 := \alpha_2{}^\frown\alpha_3{}^\frown\alpha_4$ is consistent with σ_2. Therefore, by definition of σ_1 and σ_2, the path π_1 is labeled with α_3 in the Σ_1-component and the path π_2 is labeled with α_4 in the Σ_2-component. So the path which corresponds to π_1 in the tree t is $\pi = \alpha_3{}^\frown\alpha_4$ and the path which corresponds to π_2 in the tree t is π as well. As both ρ_1 and ρ_2 are accepting, each node on π is labeled with $(\top, \top)$.

Now by construction of $\mathcal{A}_1$, this yields, that only states from $Q^{\mathcal{S}_1}$ occur in the Q^1-component of π_1 and the infinite sequence $\rho_{\mathcal{S}} = (q_j)_{j\in\mathbb{N}} \in (Q^{\mathcal{S}_1})^\omega$ of states of $\mathcal{S}_1$ constitutes a run of $\mathcal{S}_1$ on the ω-word from $(\Sigma^{p_1})^\omega$ which is obtained from the corresponding components of the labels of π_1. Now this ω-word is presicely β_1 and since ρ_1 is accepting, so is $\rho_{\mathcal{S}}$. Hence, $\beta_1 \in L_{p_1}$ and in the same way we obtain $\beta_2 \in L_{p_2}$.

Now let conversely $\sigma = (\sigma_1, \sigma_2)$ be a joint strategy for p_1 and p_2. We define the tree $t : (\Sigma_1 \times \Sigma_2) \to \mathbb{B}^2$ as follows. For $u{}^\frown v \in (\Sigma_1 \times \Sigma_2)^*$ let $t(u{}^\frown v) = (\top, \zeta)$ if, and only if, there is some $w \in \Gamma_1^{|u|}$ such that $w{}^\frown v{}^\frown u$ is consistent with σ_1 and let $t(u{}^\frown v) = (\zeta, \top)$ if, and only if, there is some $w \in \Gamma_2^{|v|}$ such that $w{}^\frown u{}^\frown v$ is consistent with σ_2. Now we define the run $\rho_1 : T^1 \to (\Sigma_1 \times \Sigma_2)^* \times Q^1$ inductively such that, for all $x \in T^1$, the unique path π from the root of ρ_1 to x is labeled with elements from $(Q^{\mathcal{S}_1} \cup \{q_{accept}\}) \times \Gamma_1^\epsilon$ in the Q^1-components and $w{}^\frown v{}^\frown u \in (\Gamma_1 \times \Sigma_2 \times \Sigma_1)^*$ is consistent with σ_1, where $(u, v) = \Pr_{(\Sigma_1\times\Sigma_2)^*}(\rho_1(x))$ and $w \in \Gamma_1^{|u|}$ is the labeling of π in the Γ_1-components (of the Q^1-components). The run ρ_2 is constructed completely analogously.

First, $\epsilon \in T^1$ and $\rho_1(\epsilon) = (\epsilon, q_0^1)$. Now let $T^1 \cap \mathbb{N}^m$ and the values of ρ_1 on this set be defined for some natural number m such that the conditions formulated above are fulfilled. Consider some $x \in T^1 \cap \mathbb{N}^m$, let $\rho_1(x) = (u{}^\frown v, (q, a))$ for some $u{}^\frown v \in (\Sigma_1 \times \Sigma_2)^m$ and some $(q, a) \in Q^1$ and let $t(u{}^\frown v) = (\zeta_1, \zeta_2)$ for some $\zeta_1, \zeta_2 \in \mathbb{B}$. Moreover, let $\sigma_1(w{}^\frown v) = b_3 \in \Sigma_1$, where $w \in \Gamma_1^m$ is the labeling of the unique path π from the root of ρ_1 to x in the Γ_1-components (of the Q^1-components). By definition of $\mathcal{A}_1$, in $\delta(((q, a), (\zeta_1, \zeta_2))$ there is a conjunct $\bigwedge_{[(b_1,b_2)\in\Gamma_1\times\Sigma_2]}(\downarrow_{(b_3,b_2)}, (q^{(b_1,b_2)}, b_1))$ and we define the set of successors of x in T^1 as $\{x \cdot i \,|\, i = 1, \ldots, r\}$, where $\Gamma_1 \times \Sigma_2 = \{(b_1^1, b_2^1), \ldots, (b_1^r, b_2^r)\}$. Moreover, $\rho_1(x \cdot i) = ((u \cdot b_3){}^\frown(v \cdot b_2), (q^{(b_1^i,b_2^i)}, b_1))$ for $i = 1, \ldots, r$.

Now we prove that, for $i = 1, \ldots, r$, the node $x \cdot i$ fulfills the conditions formulated above. So let $i \in \{1, \ldots, r\}$ and denote $(b_1^i, b_2^i) = (b_1, b_2)$. As $w{}^\frown v{}^\frown u$ is consistent with σ_1, by definition of b_3, $(w \cdot b_1){}^\frown(v \cdot b_2){}^\frown(u \cdot b_3)$ is also consistent with σ_1. For the proof that $q^{(b_1,b_2)} \neq q_{reject}$, first notice that $q \neq q_{reject}$. Moreover, as $w{}^\frown v{}^\frown u$ is consistent with σ_1, by definition of t we have $t(u{}^\frown v) = (\top, \zeta)$ for some $\zeta \in \mathbb{B}$, that means, $\zeta_1 \neq \bot$. Now we consider the remaining two cases. If

$q \in Q^{\mathcal{S}_1}$ and $\zeta_1 = \zeta_2 = \top$, then $q^{(b_1,b_2)} = \delta^{\mathcal{S}_1}(q,(b_1,b_2,b_3)) \in Q^{\mathcal{S}_1}$. If $q = q_{accept}$ or $q \in Q^{\mathcal{S}_1}$ and additionally $\zeta_1 = \top$ and $\zeta_2 = \bot$ then $q' = q_{accept}$ for $i = 1, \dots, r$.

Obviously, by construction, ρ_1 is a run of $\mathcal{A}_1$ on t. To prove that ρ_1 is accepting, let π_1 be some infinite path through ρ_1. Then, by construction of ρ_1, $\alpha_1{\restriction_k}^\frown\alpha_4{\restriction_k}^\frown\alpha_3{\restriction_k}$ is consistent with σ_1 for all $k \in \mathbb{N}$, where $\alpha_1{}^\frown\alpha_4{}^\frown\alpha_3 \in (\Gamma_1 \times \Sigma_2 \times \Sigma_1)^\omega$ is the labeling of π_1 in the $\Gamma_1 \times \Sigma_2 \times \Sigma_1$-components. Hence, $\alpha_1{}^\frown\alpha_4{}^\frown\alpha_3$ is consistent with σ_1.

Now first, by construction of ρ_1, we have $\Pr_{Q^1}(\pi_1{\restriction_k}) \in (Q^{\mathcal{S}_1} \cup \{q_{accept}\}) \times \Gamma_1^\epsilon$. Moreover, if $\Pr_{Q^1}(\pi_1{\restriction_k}) \in \{q_{accept}\} \times \Gamma_1^\epsilon$ for some $k \in \mathbb{N}$, then π_1 is accepting, so assume that $\Pr(Q^1)(\pi_1{\restriction_k}) \in Q^{\mathcal{S}_1} \times \Gamma_1^\epsilon$ for all $k \in \mathbb{N}$. Then $t(\alpha_3{\restriction_k}^\frown\alpha_4{\restriction_k}) = (\top,\top)$ for all $k \in \mathbb{N}$ so by construction of t, for all $k \in \mathbb{N}$, there is some $w \in \Gamma_2^*$ such that $w{}^\frown\alpha_3{\restriction_k}{}^\frown\alpha_4{\restriction_k}$ is consistent with σ_2. Therefore, by König's Lemma, there is some $\alpha_2 \in \Gamma_2^\omega$ such that $\alpha_2{}^\frown\alpha_3{}^\frown\alpha_4$ is consistent with σ_2 and as $\alpha_1{}^\frown\alpha_4{}^\frown\alpha_3$ is consistent with σ_1, $\alpha = \alpha_1{}^\frown\alpha_2{}^\frown\alpha_3{}^\frown\alpha_4$ is consistent with σ. Since σ is winning, $\alpha_1{}^\frown\alpha_4{}^\frown\alpha_3 \in L_{p_1}$, so the unique run of $\mathcal{S}_1$ on $\alpha_1{}^\frown\alpha_4{}^\frown\alpha_3$ is accepting. As, by the definition of $\mathcal{A}_1$, this run coincides with the labeling in the $Q^{\mathcal{S}_1}$-components of π_1, the path π_1 is accepting. As π_1 was chosen arbitrarily, ρ_1 is accepting. Completely analogously, one shows that ρ_2 is an accepting run of $\mathcal{A}_2$ on t. Hence, $t \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. □

4.2 Undecidability

First, if there are at least two connected processes which may have a deterministic 1-counter specification, then those two processes can directly simulate a 2-register machine. As the halting problem for such machines is undecidable, we obtain the following result.

Theorem 13. *The realizability problem is undecidable for any connected architecture with at least two local deterministic 1-counter specifications.*

Proof. We proceed by a reduction from the halting problem for 2-register machines. For this, let $\mathcal{R}$ be a 2-register machine consisting of a sequence I_0, …, I_{k-1}, $I_k = \texttt{stop}$ of instructions $I_j \in \{\texttt{inc}(R_i), \texttt{dec}(R_i), \texttt{if } R_i = 0 \texttt{ goto } l \mid i \in [2], l \in [k]\}$ with the usual decrease and increase operations.

Let $p_{i_0}, \dots, p_{i_m}$ be processes such that p_{i_0} and p_{i_m} have pushdown specifications and such that, for all $j \in [m]$, p_{i_j} sends information to $p_{i_{j+1}}$ or vice versa. Moreover, we choose a set channels $C_m = \{c_j | j \in [m]\}$ such that, for all $j \in [m]$, $c_j \in C_{p_j}$ and $r(c_j) = p_{j+1}$ or vice versa. We define $\Sigma_c = [k]$ for all $c \in C_m$ and we silence all other channels $c \notin C_m$ by defining $\Sigma_c = \{\sharp\}$.

For a process p_j with $0 < j < m$ we define $L_{p_j} = \{\alpha \in \Sigma_{\text{in}}^{p_j} \times \Sigma_{\text{out}}^{p_j} | \Pr_{\Sigma_{c_j}}(\alpha) = \Pr_{\Sigma_{c_{j+1}}}(\alpha)\}$. So, if the processes have a joint winning strategy, the information sent via c_0 and c_m has to be identical in each step.

The specifications $L_{p_{i_0}}$ and $L_{p_{i_m}}$ are given by deterministic 1-counter automata which recognize them. We define $\mathcal{P}_0 = (Q, \Sigma, \{A\}, q_{\text{in}}, \delta, \bot, \text{col})$ as follows, where we consider only the component from Σ_{c_0}. $Q = \{q_i \mid i \in [k]\} \cup \{q_{believe}, q_{accept}, q_{reject}\}$, $q_{\text{in}} = q_0$, $\Sigma = [k]$ and $\text{col}(q) = 1$ for all $q \in Q \setminus \{q_{accept}\}$

and $\mathrm{col}(q_{accept}) = 0$. We define the transition function δ by a case distinction. For any $i \in [k]$

- if $I_i \in \{\mathtt{inc}(R_1), \mathtt{dec}(R_1)\}$, $\delta(q_i, i, Z) = \delta(q_{believe}, i, Z) = (q_{i+1}, Z)$
- if $I_i = \mathtt{if}\ R_1 = 0\ \mathtt{goto}\ l$, $\delta(q_i, i, Z) = \delta(q_{believe}, i, Z) = (q_{believe}, Z)$
- if $I_i = \mathtt{inc}(R_0)$, $\delta(q_i, i, Z) = \delta(q_{believe}, i, Z) = (q_{i+1}, AZ)$
- if $I_i = \mathtt{dec}(R_0)$, $\delta(q_i, i, Z) = \delta(q_{believe}, i, Z) = (q_{i+1}, \epsilon)$
- if $I_i = \mathtt{if}\ R_0 = 0\ \mathtt{goto}\ l$,
  $\delta(q_i, i, A) = \delta(q_{believe}, i, A) = (q_{i+1}, A)$ and
  $\delta(q_i, i, \bot) = \delta(q_{believe}, i, \bot) = (q_l, \bot)$
- if $I_i = \mathtt{stop}$, $\delta(q_i, i, Z) = \delta(q_{believe}, i, Z) = (q_{accept}, Z)$
- $\delta(q_i, j, Z) = (q_{reject}, Z)$ if $j \neq i$
- $\delta(q, i, Z) = (q, Z)$ for $q \in \{q_{accept}, q_{reject}\}$

The automaton $\mathcal{P}_1$ is defined in a completely analogous way with registers R_0 and R_1 swapped. Then obviously a sequence $\gamma \in Q^\omega$ is in $L(\mathcal{P}_0) \cap L(\mathcal{P}_1)$ if, and only if, the run of $\mathcal{R}$ which is uniquely determined by this sequence of states (and the empty registers at the beginning) is finite. □

The next result restricts the decidable architectures which may have one deterministic 1-counter specification.

Theorem 14. *The realizability problem is undecidable for any architecture with two connected processes $p \neq p'$ such that p is reachable and has a deterministic 1-counter specification and p' is not better informed than p.*

Proof. We proceed by a reduction from the halting problem for 2-register machines. For this, let $\mathcal{R}$ be a 2-register machine as in the proof of Theorem 13.

First, we show the result for an architecture consisting only of the two system processes p, p' where p_{env} sends information to p via some channel $c_0 \in C_{p_{env}}$ and p and p' communicate via some channel c_1. Moreover, p' has some channel $c_2 \in C_{p'}$ which may be c_1 or, in case $c_1 \in C_p$, $c_2 \neq c_1$. Roughly, the local specification of p' requires that p' writes a sequence of configurations of $\mathcal{R}$ and that its output symbol equals the input symbol that it receives from p in each step. Notice that those symbols have to be chosen simultaneously. In each step, the environment may trigger the 1-counter automaton recognizing L_p to check whether the next two configurations are in the successor relation w.r.t. one of the registers. For this, the environment has two special symbols s_j for $j \in \{1, 2\}$ where j determines which of the registers should be checked. Obviously, the system processes have a joint winning strategy iff $\mathcal{R}$ does not halt.

To be more formal, we start by defining the alphabets for the channels as $\Sigma_{c_0} = \{\sharp, 0, 1\}$ and $\Sigma_{c_1} = \Sigma_{c_2} = [k] \cup \{A_0, A_1\}$ and we define $L_{p'} = \{\alpha \in (\Sigma^{p'})^\omega | \mathrm{Pr}_{\Sigma_{c_1}}(\alpha) = \mathrm{Pr}_{\Sigma_{c_2}}(\alpha)\}$ and $L_p = \{\alpha \in (\Sigma^p)^\omega | \mathrm{Pr}_{(\Sigma_{c_0} \times \Sigma_{c_1})}(\alpha) \in L\}$ where $L \subseteq \Sigma_{c_0} \times \Sigma_{c_1}$ is the deterministic 1-counter language defined as follows.

$\alpha = \alpha_0 \frown \alpha_1 \in L$ if, and only if either $\alpha_0 = \sharp^\omega$ and α_1 does not contain the symbol $k - 1$ or the following conditions hold.

– α_1 has the form $C_0C_1\ldots$ where $C_i \in [k]\cdot\{A_0\}^*\cdot\{A_1\}^*$ for each i and $C_0 = 0$, that is, C_i represents a configuration of $\mathcal{R}$ and C_0 represents the initial configuration with empty registers. Moreover, for any $i \in \mathbb{N}$, if $C_i = \lambda A_0^{x_0}A_1^{x_1}$ then $\lambda \neq k-1$.
– α_0 has the form $\sharp^{i-1}\mathbf{s}\beta$ with $\beta \in \Sigma_{c_0}^\omega$ and for the smallest $j \geq i$ such that $\alpha_1(j) \in [k]$ the following holds. If $\alpha_1 = \alpha_1\!\upharpoonright_j C_xC_y\beta$ where $C_x = \lambda_x A_0^{x_0}A_1^{x_1}$, $C_y = \lambda_y A_0^{y_0}A_1^{y_1}$ and $\beta \in \Sigma_{c_1}^\omega$ and $C_z = \lambda_z A_0^{z_0}A_1^{z_1}$ is the successor configuration of C_x, then $\lambda_y = \lambda_z$ and $A_{\mathbf{s}}^{y_{\mathbf{s}}} = A_{\mathbf{s}}^{z_{\mathbf{s}}}$

If $\mathcal{R}$ does not halt if started with initially empty registers then p and p' have the following joint winning strategy. They write an infinite sequence of configurations of $\mathcal{R}$ into c_1 and c_2 simultaneously which represents the run of $\mathcal{R}$. Obviously, both local specifications will be fulfilled.

If, on the other hand, $\mathcal{R}$ halts if started with empty registers, then let σ be any joint strategy of p and p'. First, the same information has to be written to c_1 and c_2 in each step.

To see how the environment can spoil σ, let $\alpha_1{}^\frown\alpha_2 \in (\Sigma_{c_1} \times \Sigma_{c_2})^\omega$ be the word produced by σ when p receives input $\alpha_0 = \sharp^\omega$ from the environment and let $\alpha_1 = \alpha_2 = C_0C_1\ldots \in \Sigma_{c_1}^\omega$. If $\alpha_0{}^\frown\alpha_1 \notin L$, then σ is not a winning strategy, so assume $\alpha_0{}^\frown\alpha_1 \in L$. Since $\mathcal{R}$ halts if started with empty registers, α_1 cannot be the run of $\mathcal{R}$ on initially empty registers. Let i be minimal, such that $C_i \not\vdash C_{i+1} = \lambda A_0^{x_0}A_1^{x_1}$ and let $C = \gamma A_0^{y_0}A_1^{y_1}$ be the successor configuration of C_i (notice that since $\alpha_0{}^\frown\alpha_1 \in L$, α_1 does not contain the symbol $k-1$, so C_i is not the halting configuration). If $x_0 \neq y_0$, define $\mathbf{s} = 0$, if $x_1 \neq y_1$, define $\mathbf{s} = 1$ and if $\lambda \neq \gamma$ define $\mathbf{s}$ arbitrary. Now let $\overline{\alpha}_1{}^\frown\overline{\alpha}_2 \in (\Sigma_{c_1} \times \Sigma_{c_2})^\omega$ be the word produced by σ when p receives input $\overline{\alpha}_0 = \sharp^{|C_0\ldots C_{i-1}|}\mathbf{s}\sharp^\omega$ from the environment. Obviously, $\overline{\alpha}_2(0) = \alpha_2(0)$ which implies $\overline{\alpha}_1(0) = \alpha_1(0)$, so $\overline{\alpha}_2(1) = \alpha_2(1)$, thus $\overline{\alpha}_1(1) = \alpha_1(1)$ and so on, hence $\overline{\alpha}_2 = \alpha_2$ and $\overline{\alpha}_1 = \alpha_1$. Since $\overline{\alpha}_0{}^\frown\overline{\alpha}_1$ is consistent with σ and $\overline{\alpha}_0{}^\frown\overline{\alpha}_1 = \overline{\alpha}_0{}^\frown\alpha_1 \notin L$, σ is not a winning strategy.

To extend the proof to the general case, we simulate the communication between p and p' as above using a sequence of channels connecting p and p'. Furthermore, the signal $\mathbf{s}$ from the environment can be transmitted to p via the (directed) path from p_{env} to p. Notice that the signal $\mathbf{s}$ will arrive at p with a delay (which depends only on the given architecture), so we have to adapt the specifications to postpone the starting point of the production of a sequence of configurations by p and p' so that p_{env} will be able to denounce already the first two configurations. Moreover, although the set of channels used for this transmission is, in general, not disjoint from the set of channels connecting p and p', by a simple adaption of the alphabets and specifications, the different information can be sent along the same channels. Finally, since p' is not better informed than p, the signal $\mathbf{s}$ can be kept away from p'. □

Notice that to apply the automata constructions from the decidability proofs of Section 4.1 to an architecture as in Theorem 14, one has to construct alternating pushdown tree automata since the process with the pushdown specification is not the one with the lowest level of information. However, nonemptiness of alternating pushdown tree automata is not decidable.

The remaining two results concern regular local specifications and we just state them here without giving proofs. Essentially, those results can be proved by combining and refining ideas from [14], [5] and [11]. Some details can be found in Appendix B.

Theorem 15. *The realizability problem for regular local specifications is undecidable for any architecture with a reachable process p_1 such that p_1 sends information to processes $p_2 \neq p_3$ which are not better informed than p_1.*

Theorem 16. *The realizability problem for regular local specifications is undecidable for any architecture with at least two incomparably informed processes p_1 and p_2 which are both reachable such that there is a process $p_3 \notin \{p_1, p_2\}$ which is reachable from both p_1 and p_2.*

5 Characterization

Now we characterize the exact classes of architectures which are decidable for local regular specifications and, for each such decidable class, we determine the exact set of processes which may have a deterministic contextfree specification such that decidability still holds. Notice that an architecture which is already undecidable for regular local specifications is clearly undecidable if we additionally allow deterministic contextfree specifications. Since for local specifications the realizability problem for some architecture is decidable if, and only if, it is decidable for every connected subarchitecture, w.l.o.g. we restrict our attention to connected architectures. We also note that in the case of local specifications the hidden channels of the environment are futile.

In the following, we denote the class of all pipelines with backward-channels by $\mathcal{K}_1$ and the class of all two-flanked pipelines with backward-channels which have either only two system processes or which do not have a backward-channel from the last process by $\mathcal{K}_2$. Moreover, for an architecture $\mathfrak{A}$, $M(\mathfrak{A})$ denotes the set of all system processes of $\mathfrak{A}$ which are not reachable. Notice that $P_{sys} \setminus M(\mathfrak{A})$ induces a subarchitecture of $\mathfrak{A}$.

Theorem 17. *Let $\mathfrak{A} = (P^{\mathfrak{A}}, C^{\mathfrak{A}}, r^{\mathfrak{A}})$ be a connected architecture. Then $\mathfrak{A}$ is decidable for regular local specifications if, and only if, any connected subarchitecture of $\mathfrak{A}(P^{\mathfrak{A}}_{sys} \setminus M(\mathfrak{A}))$ is in $\mathcal{K}_1 \cup \mathcal{K}_2$. Moreover, $\mathfrak{A}$ remains decidable for deterministic contextfree specifications if, and only if, one of the following conditions hold.*

(1) $\mathfrak{A} \in \mathcal{K}_1$ and $L_{p_1}, \ldots, L_{p_{n-1}}$ are regular.
(2) There is a $p \in M(\mathfrak{A})$ such that $L_{p'}$ is regular for all $p' \in P^{\mathfrak{A}}_{sys} \setminus \{p\}$.

Proof. We prove here only that $\mathfrak{A}$ is decidable, if any connected subarchitecture of $\mathfrak{A}(P^{\mathfrak{A}}_{sys} \setminus M(\mathfrak{A}))$ is in $\mathcal{K}_1 \cup \mathcal{K}_2$ (see Figure 2) and there is a $p \in M(\mathfrak{A})$ such that $L_{p'}$ is regular for all $p' \in P_{sys} \setminus \{p\}$. For this, we define $M := M(\mathfrak{A})$ and $C_M := \bigcup_{p \in M} C_p$ and we consider some labeling $(\Sigma_c)_{c \in C}$ of $\mathfrak{A}$.

First, let $\tilde{L}_p = \{\alpha \in \Sigma_M^\omega \mid \Pr_{\Sigma^p}(\alpha) \in L_p\}$ for $p \in M$ and let $\tilde{L}_M = \bigcap_{p \in M} \tilde{L}_p$. Then $\alpha_M \in \tilde{L}_M$ iff there is a joint strategy $\sigma_M = (\sigma_p)_{p \in M}$ for the processes in M such that any global system behavior β of $\mathfrak{A}$ which is consistent with σ_M fulfills all local specifications of the processes in M and $\Pr_{\Sigma_M}(\beta) = \alpha_M$. Moreover, as at most one specification L_p for $p \in M$ is deterministic contextfree and all others are regular, $\tilde{L}_M$ is deterministic contextfree.

Now consider any connected subarchitecture $\mathfrak{B} = (P, C, r)$ of $\mathfrak{A}(P_{sys} \setminus M)$ and let $C_{M \to \mathfrak{B}} = C_M \cap (r^{\mathfrak{A}})^{-1}(P)$. As all specifications L_p for $p \in P_{sys}^{\mathfrak{A}} \setminus M$ are regular, it suffices to consider the case where $\mathfrak{B}$ is a two-flanked pipeline with backward-channels and here, we only consider the case where $\mathfrak{B}$ has no backward-channel from the last process. We define the architecture $\widehat{\mathfrak{B}} = (P, \widehat{C}, \widehat{r})$ as follows.

The channels from M to P are simulated by new channels $C_{M \to \mathfrak{B}} \subseteq \widehat{C}$ of process p_{n-1}, via which p_{n-1} sends information to the respective receipients of the original channels. Moreover, p_{n-1} has a set of duplicate channels $C^d_{M \to \mathfrak{B}}$ which are read by process p_n and the specification $\widehat{L}_{p_{n-1}}$ requires that the information sent along the channels $C_{M \to \mathfrak{B}}$ is the same as the information sent along the channels $C^d_{M \to \mathfrak{B}}$. Apart from this requirement, the specifications of the processes are just as before, i.e, $\widehat{L}_{p_i} = L_{p_i}$ for $i = 1, \ldots, n-2$, $\widehat{L}_{p_n}$ is L_{p_n}, adapted to the new channels (on which $\widehat{L}_{p_n}$ does not impose any conditions) and $\widehat{L}_{p_{n-1}}$ is $L_{p_{n-1}}$ together with the additional requirement described above. For $i = 1, \ldots, n-1$, by $\widehat{\Sigma}_i$ we denote the alphabet which labels all the channels from process p_i to process p_{i+1} in the new architecture $\widehat{\mathfrak{B}}$ (notice that this labeling is uniquely determined by the definition of $\widehat{\mathfrak{B}}$).

Now let $\mathcal{N}_{n-1}$ be the parity NTA over $\mathbb{B}$-labeled $\widehat{\Sigma}_{\text{out}}^{\geq n-1}$-trees according to Lemma 5, applied to $\widehat{\mathfrak{B}}$. Then there is a parity NTA $\mathcal{N}'_{n-1}$ over $\mathbb{B}$-labeled $\widehat{\Sigma}_{n-1}$-trees which accepts a tree t iff $\text{wide}(t, \widehat{\Sigma}_{\text{out}}^{b,p_{n-1}}) \in L(\mathcal{N}_{n-1})$. Moreover, let $\mathcal{N}_n$ be the parity NTA over $\mathbb{B}$-labeled $\widehat{\Sigma}_{n-1}$-trees according to Lemma 10, applied to $\widehat{\mathfrak{B}}$. Then there is a parity NTA $\mathcal{N}^{\mathfrak{B}}$ over $\mathbb{B}$-labeled $\widehat{\Sigma}_{n-1}$-trees such that $L(\mathcal{N}^{\mathfrak{B}}) = L(\mathcal{N}'_{n-1}) \cap L(\mathcal{N}_n)$. Furthermore, we construct an alternating parity automaton $\mathcal{A}^{\mathfrak{B}} = (Q, \Sigma^{\bar{p}}_{M,d}, \delta, q_0, \text{col})$ such that $\mu \in L(\mathcal{A}^{\mathfrak{B}})$ iff there is a tree $t \in \mathbb{T}_C(\widehat{\Sigma}_{n-1})$ such that $\Pr_{\Sigma^d_{M \to \mathfrak{B}}}(L^\omega(t)) = \{\mu\}$ and $t \in L(\mathcal{N}^{\mathfrak{B}})$.

- $Q = Q^{\mathcal{N}} \times \mathbb{B} \times (\widehat{\Sigma}_{n-1})_\epsilon$
- $q_0 = (q_0^{\mathcal{N}}, \top, \epsilon)$
- $\text{col}(q, \zeta, b) = \text{col}(q)$ for all $q \in Q^{\mathcal{N}}$, all $\zeta \in \mathbb{B}$ and all $b \in (\widehat{\Sigma}_{n-1})_\epsilon$
- for $(q, \top, a_1) \in Q$ and $b_1 \in \Sigma^{\bar{p}}_{M,d}$

$$\delta((q, \top, a_1), b_1) = \bigvee_{[\emptyset \neq X \subseteq \widehat{\Sigma}_{n-1}]} \bigvee_{[\phi \in \delta^{\mathcal{N}}(q, \top)]} \bigwedge_{[a_2 \in \widehat{\Sigma}_{n-1}]} (\downarrow_{a_2}, q^{(X, \phi, a_2, b_1)})$$

where

$$q^{(X,\phi,a_2,b_1)} = \begin{cases} (p, \top, a_2) & \text{if } a_2 \in X \text{ and } \Pr_{\Sigma^{\bar{p}}_{M,d}}(a_2) = b_1 \text{ and } (\downarrow_{a_2}, p) \in \phi \\ (p, \bot, a_2) & \text{if } (a_2 \notin X \text{ or } \Pr_{\Sigma^{\bar{p}}_{M,d}}(a_2) \neq b_1) \text{ and } (\downarrow_{a_2}, p) \in \phi \end{cases}$$

– for $(q, \bot, a_1) \in Q$ and $b_1 \in \Sigma^{\bar{p}}_{M,d}$

$$\delta((q, \bot, a_1), b_1) = \bigvee_{[\phi \in \delta^{\mathcal{N}}(q, \top)]} \bigwedge_{[a_2 \in \widehat{\Sigma}_{n-1}]} (\downarrow_{a_2}, q^{(\phi, a_2)})$$

where $q^{(\phi,a_2)} = (p, \bot, a_2)$ if $(\downarrow_{a_2}, p) \in \phi$

Now consider some $\mu \in L(\mathcal{A}^{\mathfrak{B}})$, that means, there is a tree $t \in \mathbb{T}_C(\widehat{\Sigma}_{n-1})$ such that $\Pr_{\Sigma^d_{M\to\mathfrak{B}}}(L^\omega(t)) = \{\mu\}$ and $t \in L(\mathcal{N}^{\mathfrak{B}})$. As $t \in L(\mathcal{N}^{\mathfrak{B}})$, there is a joint winning strategy $\widehat{\sigma} = (\widehat{\sigma}_1, \ldots, \widehat{\sigma}_n)$ for the processes $p_1, \ldots, p_n$ such that the language generated by $\widehat{\sigma}_1 \circ \Pr_{\widehat{\Sigma}_1} \circ \ldots \circ \widehat{\sigma}_{n-1} \circ \Pr_{\widehat{\Sigma}_{n-1}}$ over $\Sigma^\omega_{\text{env}}$ is a subset of $L^\omega(t)$. So any global system behavior $\widehat{\alpha}$ of $\widehat{\mathfrak{A}}$ which is consistent with $\widehat{\sigma}$ fulfills $\Pr_{\Sigma^d_{M\to\mathfrak{B}}}(\widehat{\alpha}) = \mu$. We define strategies $\sigma_1, \ldots, \sigma_n$ for the processes $p_1, \ldots, p_n$ in the architecture $\mathfrak{A}$ as follows. First, $\sigma_i = \widehat{\sigma}_i$ for $i = 1, \ldots, n-2$ and $\sigma_{n-1} = \widehat{\sigma}_{n-1} \circ \Pr_{\Sigma^{p_{n-1}}_{\text{out}}}$. Moreover, $\sigma_n(u) = \widehat{\sigma}_n(u^\frown \mu\!\uparrow_{|u|})$ for all $u \in (\Sigma^{p_n}_{\text{in}})^*$.

Now consider any global system behavior β of $\mathfrak{A}$ which is consistent with $\sigma = (\sigma_1, \ldots, \sigma_n)$ and fulfills $\Pr_{\Sigma_{M\to\mathfrak{B}}}(\beta) = \mu$. We define $\widehat{\alpha} = \Pr_A(\beta)^\frown \mu$ where $A = \prod_{c \in \widehat{C} \cap C^{\mathfrak{A}}} \Sigma_c$. Then $\Pr_{\Sigma^{p_i}}(\widehat{\alpha}) = \Pr_{\Sigma^{p_i}}(\beta)$ for $i = 1, \ldots, n-2$, so by definition of the strategies $\sigma_1, \ldots, \sigma_{n-2}$ the global system behavior $\widehat{\alpha}$ of $\widehat{\mathfrak{B}}$ is consistent with $\widehat{\sigma}_1, \ldots, \widehat{\sigma}_{n-2}$. Moreover, $\Pr_{\Sigma^d_{M\to\mathfrak{B}}}(\widehat{\alpha}) = \Pr_{\Sigma_{M\to\mathfrak{B}}}(\widehat{\alpha})$, so as $\widehat{\sigma}$ is a joint winning strategy, by definition of σ_{n-1} and $\widehat{L}_{p_{n-1}}$ we have that $\widehat{\alpha}$ is consistent with $\widehat{\sigma}_{n-1}$. Finally, by definition of σ_n, $\widehat{\alpha}$ is consistent with $\widehat{\sigma}_n$. As $\widehat{\sigma}$ is a joint winning strategy, $\Pr_{\widehat{\Sigma}^{p_i}}(\widehat{\alpha}) \in \widehat{L}_{p_i}$ for $i = 1, \ldots, n$ and thus $\Pr_{\Sigma^{p_i}}(\beta) \in L_{p_i}$ for $i = 1, \ldots, n$.

So we have shown that, if $\mu \in L(\mathcal{A}^{\mathfrak{B}})$, then there is a strategy $\sigma = (\sigma_1, \ldots, \sigma_n)$ for processes $p_1, \ldots, p_n$ such that any global system behavior β of $\mathfrak{A}$ which is consistent with σ and fulfills $\Pr_{\Sigma_{M\to\mathfrak{B}}}(\beta) = \mu$ fulfills all local specifications of the processes $p_1, \ldots, p_n$. Using this, one can furthermore show that $\mu \in L(\mathcal{A}^{\mathfrak{B}})$ if, and only if, there is a strategy $\sigma = (\sigma_1, \ldots, \sigma_n)$ for processes $p_1, \ldots, p_n$ in the architecture $\mathfrak{A}$ such that any global system behavior β of $\mathfrak{A}$ which is consistent with σ and fulfills $\Pr_{\Sigma_{M\to\mathfrak{B}}}(\beta) = \mu$, fulfills all local specifications of the processes $p_1, \ldots, p_n$.

Finally, for any connected subarchitecture $\mathfrak{B}$ of $\mathfrak{A}(P_{sys} \setminus M)$ let $\tilde{L}(\mathcal{A}^{\mathfrak{B}}) = \{\alpha \in \Sigma^\omega_M \mid \Pr_{\Sigma_{M\to\mathfrak{B}}}(\alpha) \in L(\mathcal{A}^{\mathfrak{B}})\}$ and $L = \bigcap_{\mathfrak{B}} \tilde{L}(\mathcal{A}^{\mathfrak{B}}) \cap \tilde{L}_M$. Then $L \neq \emptyset$ iff the system processes have a joint winning strategy. As L is deterministic contextfree, emptiness of L can be decided. □

References

1. J.R. Büchi and L.H. Landweber. Solving Sequential Conditions by Finite-State Strategies. *Trans. Amer. Math. Soc.*, 138 (1969), pages 367- 378.
2. A. Church. Applications of Recursive Arithmetic to the Problem of Circuit Synthesis. *Sum. of the Sum. Inst. of Symb. Log.*, N.Y. 1957, Volume I, pages 3-50.

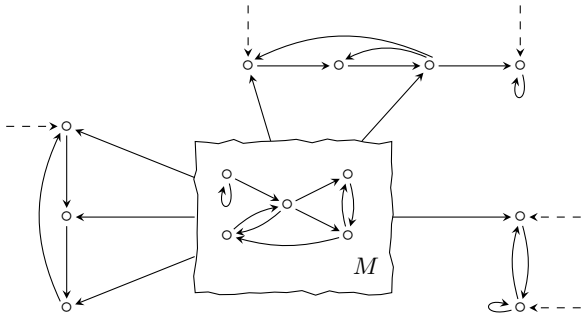

Fig. 2. Generic decidable architecture with non-reachable processes M

3. R. S. Cohen and A. Y. Gold. Theory of Omega-Languages. I. Characterizations of Omega-Context-Free Languages. *J. Comput. Syst. Sci.*, 15(2), 1977.
4. L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent Reachability Games. In *FOCS*, 1998.
5. B. Finkbeiner and S. Schewe. Uniform Distributed Synthesis. In *LICS '05*, pages 321–330. IEEE, 2005.
6. O. Finkel. Topological Properties of Omega Context-Free Languages. *Theor. Comput. Sci.*, 262(1), 2001.
7. W. Fridman and B. Puchala. Distributed Synthesis for Regular and Contextfree Specifications. In *36th International Symposium on Mathematical Foundations of Computer Science, MFCS 2011*. Springer, 2011.
8. O. Kupferman, N. Piterman, and M. Y. Vardi. Pushdown Specifications. In *LPAR*, 2002.
9. O. Kupferman and M. Y. Vardi. Church's Problem Revisited. *Bulletin of Symbolic Logic*, 5(2), 1999.
10. O. Kupferman and M. Y. Vardi. Synthesizing Distributed Systems. In *LICS*, 2001.
11. P. Madhusudan and P.S. Thiagarajan. Distributed Controller Synthesis for Local Specifications. In *ICALP '01*, pages 396–407. Springer, 2001.
12. A. Pnueli. The Temporal Logic of Programs. In *FOCS '77*. IEEE, 1977.
13. A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *POPL '89*, pages 179–190, 1989.
14. A. Pnueli and R. Rosner. Distributed Reactive Systems are Hard to Synthesize. In *FOCS '90*, pages 746–757. IEEE, 1990.
15. M.O. Rabin. Automata on Infinite Objects and Churchs Problem. Amer. Math. Soc., Providence RI, 1972.
16. S. Safra. On the Complexity of Omega-Automata. In *FOCS*, 1988.
17. I. Walukiewicz. Pushdown Processes: Games and Model Checking. In *CAV '96*, pages 62–74. Springer, 1996.

A Proofs of Section 4.1

In this section, we give a technical proof of Lemma 5. We devide the proof of into two lemmata, one for the base case and one for the induction step. First, we give the lemma for the base case as well as some technical details of the proof.

Lemma 18. *There is a parity ATA $\mathcal{A}$ over $\mathbb{B}$-labeled $\Sigma_{out}^{\geq 1}$-trees which accepts a tree $t_{out} \in \mathbb{T}_C(\Sigma_{out}^{\geq 1})$ if, and only if, there is a tree $t \in t_{in} \hookrightarrow t_{out}$ where $t_{in}(u) = \top$ for all $u \in \Sigma_0^*$ such that $\sigma(t)$ is locally winning on $L^\omega(t_{in}) = \Sigma_0^\omega$.*

Proof. We abbreviate $\Sigma = \Sigma_0$ and $\Sigma' = \Sigma_{\text{out}}^{\geq 1}$. Let $\mathcal{S} = (Q^{\mathcal{S}}, \Sigma \times \Sigma', \delta^{\mathcal{S}}, q_0^{\mathcal{S}}, \text{col})$ be a parity automaton with $L(\mathcal{S}) = \{\alpha \in (\Sigma \times \Sigma')^\omega | \text{Pr}_{\Sigma_{p_1}}(\alpha) \in L_{p_1}\}$.

We define the alternating parity tree automaton $\mathcal{A} = (Q, \mathbb{B}, \delta, q_0, \text{col})$ as follows.

- $Q = ((Q^{\mathcal{S}} \cup \{q_{accept}\}) \times \mathbb{B} \times \Sigma_\epsilon) \cup \{q_{reject}\}$
- $q_0 = (q_0^{\mathcal{S}}, \top, \epsilon)$
- $\text{col}(q, \zeta, a) = \text{col}^{\mathcal{S}}(q)$ for $(q, \zeta, a) \in (Q^{\mathcal{S}} \cup \{q_{accept}\}) \times \mathbb{B} \times \Sigma_\epsilon$ where we define $\text{col}^{\mathcal{S}}(q_{accept}) := 0$ and $\text{col}(q_{reject}) = 1$
- for $q \in Q^{\mathcal{S}}$ and $a \in \Sigma_\epsilon$,

$$\delta((q, \top, a), \top) = \bigvee_{[b \in \Sigma']} \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_y, q^{(b,x,y)})$$

  where

$$q^{(b,x,y)} = \begin{cases} (\delta^{\mathcal{S}}(q, (x, y)), \top, x) & \text{if } y = b \\ (q_{accept}, \bot, x) & \text{if } y \neq b \end{cases}$$

- for $q \in Q^{\mathcal{S}}$, $a \in \Sigma_\epsilon$ and $\zeta \in \mathbb{B}$, $\delta((q_{accept}, \bot, a), \zeta) = \bigwedge_{(x,y) \in \Sigma \times \Sigma'} (\downarrow_y, (q_{accept}, \bot, x))$
- for $q \in Q^{\mathcal{S}}$ and $a \in \Sigma_\epsilon$, $\delta((q, \top, a), \bot) = \bigwedge_{y \in \Sigma'} (\downarrow_y, q_{reject})$
- for $\zeta \in \mathbb{B}$, $\delta(q_{reject}, \zeta) = \bigwedge_{y \in \Sigma'} (\downarrow_y, q_{reject})$

Now we prove that the automaton is correct. We only consider the direction from left to right, the converse direction can be proved using similar arguments. So, let $t_{\text{out}} \in L(\mathcal{A})$ and let $\rho : T \to (\Sigma')^* \times Q$ with $T \subseteq \mathbb{N}^*$ be an accepting run of $\mathcal{A}$ on t_{out}.

First, by induction on m, one can easily prove that for any $u^\frown v \in (\Sigma \times \Sigma')^m$ there is exactly one $k \in T$ which corresponds to $u^\frown v$, that means, the unique path from the root of T to k is labeled with u in the Σ_ϵ-components (of the Q-components) and with v in the Σ'-components or, more precisely, $\rho(k) = (v, q)$ and $\text{Pr}_\Sigma(\text{Pr}_Q(\rho(\pi(0)) \dots \rho(\pi(|k| - 1)))) = u$.

Therefore, ρ yields the $\mathbb{B}$-labeled $\Sigma \times \Sigma'$-tree t with $t(u^\frown v) = \text{Pr}_{\mathbb{B}}(\text{Pr}_Q(\rho(k)))$ for the unique k which corresponds to $u^\frown v$. Moreover, we define the $\mathbb{B}$-labeled Σ-tree t_{in} by $t_{\text{in}}(u) = \top$ for all $u \in \Sigma^*$.

First we show that $t \in t_{\text{in}} \hookrightarrow t_{\text{out}}$. Notice that, by definition of $\mathcal{A}$, $t \in \mathbb{T}_C(\Sigma \times \Sigma')$. As $\mathcal{A}$ immediately rejects a tree, if it encounters a $\bot$-symbol in t_{out} when having guessed a $\top$-symbol for t, we have that if $t_{\text{out}}(v) = \bot$, then $t(u^\frown v) = \bot$ for all $u \in \Sigma^*$. Moreover, by definition of t_{in}, if $t_{\text{in}}(u) = \bot$, then $t(u^\frown v) = \bot$ for all $v \in (\Sigma')^*$. Therefore, condition (S1) holds.

We show conditions (S2) and (S3) simultaneously by induction on $|u|$. For $u = \epsilon$, condition (S2) is obvious. Moreover, according to the definition of $\mathcal{A}$, there is some set $S = \{(\downarrow_{y_1}, q^{(b,x_1,y_1)}), \ldots, (\downarrow_{y_r}, q^{(b,x_r,y_r)})\} \in \delta((q_0, \top, \epsilon), \top)$ for some $b \in \Sigma'$ such that $q^{(b,x_l,y_l)} \in (Q^{\mathcal{S}} \cup \{q_{accept}\}) \times \mathbb{B} \times \{x_l\}$ and the set of successors of ϵ in T is precisely $\{1, \ldots, r\}$ where $\rho(l) = (y_l, q^{(b,x_l,y_l)})$. By definition of $q^{(b,x_l,y_l)}$, $\Pr_{\mathbb{B}}(\Pr_Q(\rho(l)) = \top$ if, and only if, $y_l = b$. Moreover, as $1, \ldots, l$ correspond precisely to the successors of ϵ in t, the definition of t yields that for all $a \in \Sigma$ we have $t(a, b) = \top$ and $t(a, c) = \bot$ for all $c \in \Sigma' \setminus \{b\}$. Hence, condition (S3) holds.

Now let $|u| \geq 1$. As $t_{\text{in}}(u^{-1}) = \top$, by induction hypothesis for (S2) there is some $\hat{v} \in (\Sigma')^*$ such that $t((u^{-1})^\frown(\hat{v})) = \top$ and by induction hypothesis for (S3) there is some $\hat{b} \in \Sigma'$ such that $t((u^{-1}a)^\frown(\hat{v}\hat{b})) = \top$ for all $a \in \Sigma$ (as $t_{\text{in}}(u^{-1}a) = \top$). Hence, $t(u^\frown(\hat{v}\hat{b})) = \top$. Now let $v \in \Sigma'$ such that $t(u^\frown v) = \top$. As $t \in \mathbb{T}_C(\Sigma \times \Sigma')$, $t((u^{-1})^\frown(v^{-1})) = \top$ and therefore the induction hypothesis for (S2) yields $v^{-1} = \hat{v}$. By induction hypothesis for (S3), $t((u^{-1}a)^\frown\hat{v}b) = \bot$ for all $b \in \Sigma' \setminus \{\hat{b}\}$ and hence, $v = \hat{v}\hat{b}$. As $t_{\text{in}}(u) = \top$ for all $u \in \Sigma^*$, condition (S3) can be proved in the induction step just as in the base case.

To verify that $\sigma(t)$ is locally winning, let $\alpha \in (\Sigma \times \Sigma')^\omega$ be a local system behavior with $\Pr_\Sigma(\alpha) \in L^\omega(t_{\text{in}})(= \Sigma^\omega)$ which is consistent with $\sigma(t)$. By definition of $\sigma(t)$, $\alpha \in L^\omega(t)$ and by definition of $\mathcal{A}$, the unique (infinite) path π in T corresponding to α is labeled with states from $Q^{\mathcal{S}}$ in the Q-components. As ρ is accepting, any such infinite state sequence is accepting and by definition of $\mathcal{A}$, any such infinite state sequence forms a run of $\mathcal{S}$ on the corresponding word from $(\Sigma \times \Sigma')^\omega$. So each such local system behavior α is in $L(\mathcal{S})$ which yields that $\sigma(t)$ is locally winning on $L^\omega(t_{\text{in}})$. □

The following lemma establishes the induction step.

Lemma 19. *For any $1 < i < n$ and any given parity NTA $\mathcal{N}$ over $\mathbb{B}$-labeled $\Sigma_{i-1} \times \Sigma_{out}^{\geq i}$-trees there is a parity ATA $\mathcal{A}$ over $\mathbb{B}$-labeled $\Sigma_{out}^{\geq i}$-trees which accepts a tree $t_{out} \in \mathbb{T}_C(\Sigma_{out}^{\geq i})$ if, and only if, there are a tree $t_{in} \in \mathbb{T}_C(\Sigma_{i-1})$ and a tree $t \in t_{in} \hookrightarrow t_{out}$ such that $t \in L(\mathcal{N})$ and $\sigma(t)$ is locally winning on $L^\omega(t_{in})$.*

Proof. We abbreviate $\Sigma = \Sigma_{i-1}$ and $\Sigma' = \Sigma_{\text{out}}^{\geq i}$. Let $\mathcal{S} = (Q^{\mathcal{S}}, \Sigma \times \Sigma', \delta^{\mathcal{S}}, q_0^{\mathcal{S}}, \text{col})$ be a parity automaton with $L(\mathcal{S}) = \{\alpha \in (\Sigma \times \Sigma')^\omega | \Pr_{\Sigma_{p_i}}(\alpha) \in L_{p_i}\}$ and let $\mathcal{N} = (Q^{\mathcal{N}}, \mathbb{B}, \delta^{\mathcal{N}}, q_0^{\mathcal{N}}, \text{col}^{\mathcal{N}})$.

We define the alternating Muller tree automaton $\mathcal{A} = (Q, \mathbb{B}, \delta, q_0, \mathcal{F})$ as follows. Notice that this automaton can be transformed into an equivalent parity tree automaton.

- $Q = ((Q^{\mathcal{S}} \cup \{q_{accept}\}) \times [Q^{\mathcal{N}} \times \mathbb{B}] \times \Sigma_\epsilon) \cup \{q_{reject}\}$

- $q_0 = (q_0^{\mathcal{S}}, [q_0^{\mathcal{N}}, \top], \epsilon)$
- $\mathcal{F}$ consists of those subsets $X \subseteq Q$ such that $\min\{\mathrm{col}^{\mathcal{S}}(\mathrm{Pr}_{Q^{\mathcal{S}}}(x)) | x \in X\}$ is even and $\min\{\mathrm{col}^{\mathcal{N}}(\mathrm{Pr}_{Q^{\mathcal{N}}}(x)) | x \in X\}$ is even, where $\mathrm{col}^{\mathcal{S}}(q_{accept}) := 0$
- for $q \in Q^{\mathcal{S}}$, $p \in Q^{\mathcal{N}}$ and $a \in \Sigma_\epsilon$,

$$\delta((q, [p, \top], a), \top) = \bigvee_{[\emptyset \neq X \subseteq \Sigma]} \bigvee_{[b \in \Sigma']} \bigvee_{[\phi \in \delta^{\mathcal{N}}(p, \top)]} \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_y, q^{(X,b,\phi,x,y)})$$

  where

$$q^{(X,b,\phi,x,y)} = \begin{cases} (\delta^{\mathcal{S}}(q, (x, y)), [r, \top], x) & \text{if } (x, y) \in X \times \{b\} \text{ and } (\downarrow_{(x,y)}, r) \in \phi \\ (q_{accept}, [r, \bot], x) & \text{if } (x, y) \notin X \times \{b\} \text{ and } (\downarrow_{(x,y)}, r) \in \phi \end{cases}$$

- for $p \in Q^{\mathcal{N}}$, $\zeta \in \mathbb{B}$ and $a \in \Sigma_\epsilon$,

$$\delta((q_{accept}, [p, \bot], a), \zeta) = \bigvee_{[\phi \in \delta^{\mathcal{N}}(p, \bot)]} \bigwedge_{[(x,y) \in \Sigma \times \Sigma']} (\downarrow_y, (q_{accept}, [p_{xy}, \bot], x))$$

  if $(\downarrow_{(x,y)}, p_{xy}) \in \phi$
- for $q \in Q^{\mathcal{S}}$, $p \in Q^{\mathcal{N}}$ and $a \in \Sigma_\epsilon$, $\delta((q, [p, \top], a), \bot) = \bigwedge_{y \in \Sigma'} (\downarrow_y, q_{reject})$
- for $\zeta \in \mathbb{B}$, $\delta(q_{reject}, \zeta) = \bigwedge_{y \in \Sigma'} (\downarrow_y, q_{reject})$

Now we prove that the automaton is correct. Again, we only consider the direction from left to right. So, let $t_{\text{out}} \in L(\mathcal{A})$ and let $\rho : T \to (\Sigma')^* \times Q$ with $T \subseteq \mathbb{N}^*$ be an accepting run of $\mathcal{A}$ on t_{out}. By definition of $\mathcal{A}$, for any $u^\frown v \in (\Sigma \times \Sigma')^*$ there is exactly one $k \in T$, such that $\rho(k) = (v, q)$ and, for the unqiue path π from the root of ρ to k, $\mathrm{Pr}_\Sigma(\mathrm{Pr}_Q(\pi)) = u$. So, ρ yields the $\mathbb{B}$-labeled $\Sigma \times \Sigma'$-tree t with $t(u^\frown v) = \mathrm{Pr}_{\mathbb{B}}(\mathrm{Pr}_Q(\rho(k)))$ for this unique k. Moreover, we define the $\mathbb{B}$-labeled Σ-tree t_{in} by $t_{\text{in}}(u) = \top$ iff there is some $v \in (\Sigma')^*$ such that $t(u^\frown v) = \top$.

First we show that $t \in t_{\text{in}} \hookrightarrow t_{\text{out}}$. As $\mathcal{A}$ immediately rejects a tree, if it encounters a $\bot$-symbol in t_{out} when having guessed a $\top$-symbol for t, we have that if $t_{\text{out}}(v) = \bot$, then $t(u^\frown v) = \bot$ for all $u \in \Sigma^*$. Moreover, by definition of t_{in}, if $t_{\text{in}}(u) = \bot$, then $t(u^\frown v) = \bot$ for all $v \in (\Sigma')^*$. Therefore, condition (S1) holds. Conditions (S2) and (S3) can be proved easily by a simultanous induction on $|u|$.

To verify the other conditions, first notice that the infinite paths of t correspond exactly to the infinite paths of ρ which are labeled with states from $Q^{\mathcal{N}}$ and as ρ is accepting, any such infinite state sequence is accepting. Since by definition of $\mathcal{A}$, all these infinite state sequences form a run of $\mathcal{N}$ on t we have $t \in L(\mathcal{N})$. Moreover, $L^\omega(t)$ contains exatly those words $\alpha \in (\Sigma \times \Sigma')^\omega$ with $\mathrm{Pr}_\Sigma(\alpha) \in L^\omega(t_{\text{in}})$ which are consistent with $\sigma(t)$ and all the corresponding paths in ρ are labeled with states from $Q^{\mathcal{S}}$. As ρ is accepting, any such infinite state sequence is accepting and by definition of $\mathcal{A}$, any such infinite state sequence forms a run of $\mathcal{S}$ on the corresponding word from $(\Sigma \times \Sigma')^\omega$, so all these words α are in $L(\mathcal{S})$ which yields, that $\sigma(t)$ is locally winning on inputs from $L^\omega(t_{\text{in}})$. $\square$

Now we put those two lemmata together which provides a full proof of Lemma 5.

Lemma 5. *For any $1 \leq i < n$ there is a parity NTA $\mathcal{N}_i$ over $\mathbb{B}$-labeled $\Sigma_{out}^{\geq i}$-trees which accepts a tree $t_{out} \in \mathbb{T}_C(\Sigma_{out}^{\geq i})$ if, and only if, there are a $\mathbb{B}$-labeled Σ_{i-1}-tree $t_{in} \in \mathbb{T}_C(\Sigma_{i-1})$, a tree $t \in t_{in} \hookrightarrow t_{out}$ and strategies $\sigma_1, \ldots, \sigma_{i-1}$ for processes $p_1, \ldots, p_{i-1}$ such that $\sigma(t)$ is locally winning on $L^\omega(t_{in})$ and*

- *$\sigma_1 \circ Pr_{\Sigma_1} \circ \ldots \circ \sigma_{i-1} \circ Pr_{\Sigma_{i-1}}$ generates a language $L \subseteq L^\omega(t_{in})$ over Σ_0^ω*
- *$(\sigma_1, \ldots, \sigma_{i-1}, \sigma(t))$ is winning for $p_1, \ldots, p_i$.*

Proof. We prove this by induction on i. The base case $i = 1$ follows immediately from Lemma 18, so let $i > 1$ and let $\mathcal{N}_{i-1}$ be a parity NTA over $\mathbb{B}$-labeled $\Sigma_{\text{out}}^{\geq i-1}$-trees according to the induction hypothesis. Then there is a parity ATA $\mathcal{A}'_{i-1}$ over $\mathbb{B}$-labeled $\Sigma_{i-1} \times \Sigma_{\text{out}}^{\geq i}$-trees which accepts a tree t if, and only if, $\text{wide}(t, \Sigma_{\text{out}}^{b,p_{i-1}}) \in L(\mathcal{N}_{i-1})$. Furthermore, $\mathcal{A}'_{i-1}$ can be transformed into an equivalent parity NTA $\mathcal{N}'_{i-1}$. Now, let $\mathcal{A}_i$ be a parity ATA according to Lemma 19, using the automaton $\mathcal{N}'_{i-1}$ and let $\mathcal{N}_i$ be a parity NTA equivalent to $\mathcal{A}_i$.

To show that $\mathcal{N}_i$ fulfills conditions of the lemma, first let $t_{\text{out}} \in L(\mathcal{N}_i)$, that means there are a $\mathbb{B}$-labeled Σ_{i-1}-tree t_{in} and a $\mathbb{B}$-labeled $\Sigma_{i-1} \times \Sigma_{\text{out}}^{\geq i}$-tree $t \in t_{\text{in}} \hookrightarrow t_{\text{out}}$ such that $t \in L(\mathcal{N}'_{i-1})$, and $\sigma(t)$ is locally winning on $L^\omega(t_{\text{in}})$.

Since $t \in L(\mathcal{N}'_{i-1})$, $s' = \text{wide}(t, \Sigma_{\text{out}}^{b,p_{i-1}}) \in L(\mathcal{N}_{i-1})$, that means, there are a $\mathbb{B}$-labeled Σ_{i-2}-tree s_{in}, a tree $s \in s_{\text{in}} \hookrightarrow s_{\text{out}}$ and strategies $\sigma_1, \ldots, \sigma_{i-2}$ for processes $p_1, \ldots, p_{i-2}$ such that $\sigma_1 \circ \text{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-2} \circ \text{Pr}_{\Sigma_{i-2}}$ generates a language $L \subseteq L^\omega(s_{\text{in}})$ over Σ_0^ω and $(\sigma_1, \ldots, \sigma_{i-2}, \sigma(s))$ is winning for $p_1, \ldots, p_{i-1}$.

We denote $\sigma(s) = (\sigma_{i-1}, \tau_i, \ldots, \tau_n)$ and $\sigma(t) = (\sigma_i, \ldots, \sigma_n)$ and we abbreviate $(\sigma_i, \ldots, \sigma_n) = \overline{\sigma}$ and $(\tau_i, \ldots, \tau_n) = \overline{\tau}$. Now the language generated by $\sigma_1 \circ \text{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-2} \circ \text{Pr}_{\Sigma_{i-2}}$ over Σ_0^ω is a subset of $L^\omega(s_{\text{in}})$ so the language generated by $\sigma_1 \circ \text{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-2} \circ \text{Pr}_{\Sigma_{i-2}} \circ \sigma_{i-1} \circ \text{Pr}_{\Sigma_{i-1}}$ over Σ_0^ω is a subset of the language generated by $\sigma_{i-1} \circ \text{Pr}_{\Sigma_{i-1}}$ over $L^\omega(s_{\text{in}})$, which is $L^\omega_{\Sigma_{i-1}}(s) \subseteq L^\omega_{\Sigma_{i-1}}(s_{\text{out}}) = L^\omega_{\Sigma_{i-1}}(t) = L^\omega(t_{\text{in}})$.

Now we show that $\overline{\sigma}(\epsilon) = \overline{\tau}(\epsilon)$ and $\overline{\sigma}(\text{Pr}_{\Sigma_{i-1}}(\sigma^*_{i-1}(u^{-1}))) = \overline{\tau}(u)$ for all $u \in L^*(s_{\text{in}}) = L^*_{\Sigma_{i-2}}(s) \subseteq \Sigma^*_{i-2}$ with $|u| \geq 1$.

First, by definition of $\sigma(s)$, for all $a \in \Sigma_{i-2}$ with $s_{\text{in}}(a) = \top$ we have $s((a, \sigma(s)(\epsilon))) = \top$ which, by definition of $s_{\text{in}} \hookrightarrow s_{\text{out}}$, implies $s_{\text{out}}(\sigma(s)(\epsilon)) = \top$. Since $s_{\text{out}} = \text{wide}(t, \Sigma_{\text{out}}^{b,p_{i-1}})$ this yields $t(\text{Pr}_{\Sigma_{i-1} \times \Sigma_{\text{out}}^{\geq i}}(\sigma(s)(\epsilon))) = \top$ so, by definition of $\sigma(t)$, $\overline{\sigma}(\epsilon) = \sigma(t)(\epsilon) = \text{Pr}_{\Sigma_{\text{out}}^{\geq i}}(\sigma(s)(\epsilon)) = \overline{\tau}(\epsilon)$.

Now let $u \in L^*_{\Sigma_{i-2}}(s)$ with $|u| \geq 1$, that means, there is some $v \in (\Sigma_{\text{out}}^{\geq i-1})^{|u|}$ with $s(u^\frown v) = \top$. By definition of $\sigma(s)$ we have $v = \sigma(s)^*(u^{-1})$ and, for all $a \in \Sigma_{i-2}$ with $s_{\text{in}}(ua) = \top$, $s(ua^\frown v\sigma(s)(u)) = \top$ which, by definition of $s_{\text{in}} \hookrightarrow s_{\text{out}}$, implies $s_{\text{out}}(v\sigma(s)(u)) = \top$. Since $s_{\text{out}} = \text{wide}(t, \Sigma_{\text{out}}^{b,p_{i-1}})$ this yields $t(\text{Pr}_{\Sigma_{i-1} \times \Sigma_{\text{out}}^{\geq i}}(v\sigma(s)(u))) = \top$ and, as $\text{Pr}_{\Sigma_{i-1}}(\sigma^*_{i-1}(u^{-1})) = \text{Pr}_{\Sigma_{i-1}}(\sigma(s)^*(u^{-1}))$, by definition of $\sigma(t)$ we have, $\overline{\sigma}(\text{Pr}_{\Sigma_{i-1}}(\sigma^*_{i-1}(u^{-1}))) = \sigma(t)(\text{Pr}_{\Sigma_{i-1}}(\sigma(s)^*(u^{-1})))$ $= \sigma(t)(\text{Pr}_{\Sigma_{i-1}}(v)) = \text{Pr}_{\Sigma_{\text{out}}^{\geq i}}(\sigma(s)(u)) = \overline{\tau}(u)$.

It remains to prove that $(\sigma_1, \ldots, \sigma_{i-2}, \sigma_{i-1}, \sigma(t))$ is winning for $p_1, \ldots, p_i$. By induction hypothesis, $(\sigma_1, \ldots, \sigma_{i-2}, \sigma(s))$ is winning for $p_1, \ldots, p_{i-1}$ which means that $(\sigma_1, \ldots, \sigma_{i-2}, \sigma_{i-1}, \overline{\tau})$ is winning for $p_1, \ldots, p_{i-1}$. Moreover, the language generated by $\sigma_1 \circ \mathrm{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-1} \circ \mathrm{Pr}_{\Sigma_{i-1}}$ over Σ_0^ω is a subset of $L^\omega(t_{\mathrm{in}})$ and σ_i is locally winning on $L^\omega(t_{\mathrm{in}})$. Furthermore, as we have shown above, $\overline{\tau}(\epsilon) = \overline{\sigma}(\epsilon)$ and $\overline{\tau}(u) = \overline{\sigma}(\mathrm{Pr}_{\Sigma_{i-1}}(\sigma_{i-1}^*(u^{-1}))$ for all $u \in L^*(s_{\mathrm{in}})$ with $|u| \geq 1$. Now, if α is some global system behavior which is consistent with $(\sigma_1, \ldots, \sigma_{i-2}, \sigma_{i-1}, \sigma(t))$ then this shows that α is also consistent with $(\sigma_1, \ldots, \sigma_{i-2}, \sigma_{i-1}, \overline{\tau})$, so all local specifications $L_{p_1}, \ldots, L_{p_{i-1}}$ are satisfied and as $\sigma(t)$ is locally winning on $L^\omega(t_{\mathrm{in}})$, L_{p_i} is satisfied as well.

Now let conversely t_{out} be a $\mathbb{B}$-labeled $\Sigma_{\mathrm{out}}^{\geq i}$-tree such that there are a $\mathbb{B}$-labeled Σ_{i-1}-tree t_{in}, a tree $t \in t_{\mathrm{in}} \hookrightarrow t_{\mathrm{out}}$ and strategies $\sigma_1, \ldots, \sigma_{i-1}$ for processes $p_1, \ldots, p_{i-1}$ such that $\sigma_1 \circ \mathrm{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-1} \circ \mathrm{Pr}_{\Sigma_{i-1}}$ generates a language $L \subseteq L^\omega(t_{\mathrm{in}})$ over Σ_0^ω and $(\sigma_1, \ldots, \sigma_{i-1}, \sigma(t))$ is winning for $p_1, \ldots, p_i$.

As $\mathcal{N}_i$ has been constructed according to Lemma 19 using the automaton $\mathcal{N}'_{i-1}$, the automaton $\mathcal{N}_i$ accepts t_{out} if there are a $\mathbb{B}$-labeled Σ_{i-1}-tree t_{in} and a tree $t \in t_{\mathrm{in}} \hookrightarrow t_{\mathrm{out}}$ such that $t \in L(\mathcal{N}'_{i-1})$ and $\sigma(t)$ is locally winning on $L^\omega(t_{\mathrm{in}})$. So all that remains to show is $t \in L(\mathcal{N}'_{i-1})$, that means, $s_{\mathrm{out}} = \mathrm{wide}(t, \Sigma_{\mathrm{out}}^{b,p_{i-1}}) \in L(\mathcal{N}_{i-1})$. Now, by induction hypothesis, to prove this it suffices to find a $\mathbb{B}$-labeled Σ_{i-1}-tree s_{in} and a tree $s \in s_{\mathrm{in}} \hookrightarrow s_{\mathrm{out}}$ such that $\sigma_1 \circ \mathrm{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-2} \circ \mathrm{Pr}_{\Sigma_{i-2}}$ generates a language $L \subseteq L^\omega(s_{\mathrm{in}})$ over Σ_0^ω and $(\sigma_1, \ldots, \sigma_{i-2}, \sigma(s))$ is winning for $p_1, \ldots, p_{i-1}$.

First, we define $s_{\mathrm{in}}(u) = \top$ if, and only if, u is in the language generated by $\sigma_1 \circ \mathrm{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-2} \circ \mathrm{Pr}_{\Sigma_{i-2}}$ over Σ_0^*. Hence, for the corresponding ω-language L^ω, we have $L^\omega = L^\omega(s_{\mathrm{in}})$. Now we define the extended local strategy $\tau : \Sigma_{i-2}^* \to \Sigma_{\mathrm{out}}^{\geq i-1}$ for process p_{i-1} as follows: $\tau(\epsilon) = (\sigma_{i-1}(\epsilon), \sigma(t)(\epsilon))$ and $\tau(u) = \big(\sigma_{i-1}(u), \sigma(t)(\mathrm{Pr}_{\Sigma_{i-1}}(\sigma_{i-1}^*(u^{-1})))\big)$ for $u \in \Sigma_{i-2}^+$.

Now if $u \in L^*(s_{\mathrm{in}})$ then $v = \mathrm{Pr}_{\Sigma_{i-1}}(\sigma_{i-1}^*(u^{-1}))$ (or $v = \epsilon$, if $u = \epsilon$) and va with $a = \mathrm{Pr}_{\Sigma_{i-1}}(\sigma_{i-1}(u))$ are both in the language generated by $\sigma_1 \circ \mathrm{Pr}_{\Sigma_1} \circ \ldots \circ \sigma_{i-1} \circ \mathrm{Pr}_{\Sigma_{i-1}}$ over Σ_0^*. Hence, $v, va \in L^*(t_{\mathrm{in}})$, so $t((va)^\frown \sigma(t)^*(v)) = \top$ and as as $s_{\mathrm{out}} = \mathrm{wide}(t, \Sigma_{\mathrm{out}}^{b,p_{i-1}})$ we have $s_{\mathrm{out}}(\mathrm{Pr}_{\Sigma_{\mathrm{out}}^{b,p_{i-1}}}(\sigma_{i-1}^*(u))^\frown (va)^\frown \sigma(t)^*(v)) = \top$. From this we conclude $\tau \restriction_{L^*(s_{\mathrm{in}})} = \sigma(s) \restriction_{L^*(s_{\mathrm{in}})}$ for the tree $s \in s_{\mathrm{in}} \hookrightarrow s_{\mathrm{out}}$ with $s(u_{\mathrm{in}}{}^\frown u_{\mathrm{out}}) = \top$ if, and only if, $u_{\mathrm{in}} = \epsilon$ (and hence $u_{\mathrm{out}} = \epsilon$) or $u_{\mathrm{in}} \in L^*(s_{in})$ and $\tau(u_{\mathrm{in}}^{-1}) = u_{\mathrm{out}}$.

Finally, by definition of $\sigma(s)$, any global system behavior which is consistent with $(\sigma_1, \ldots, \sigma_{i-2}, \sigma(s))$ is also consistent with $(\sigma_1, \ldots, \sigma_{i-1}, \sigma(t))$. So, as $(\sigma_1, \ldots, \sigma_{i-1}, \sigma(t))$ is winning for $p_1, \ldots, p_{i-1}$, $(\sigma_1, \ldots, \sigma_{i-2}, \sigma(s))$ is winning for $p_1, \ldots, p_{i-1}$ as well. □

B Proofs of Section 4.2

Now we give some details of the proof of Theorem 15. For this, we proof undecidability of two special cases. First we consider $\mathfrak{A}_0 = (P, C, r)$ with

- $P = \{p_0, p_1, p_2, p_3\}$

– $C = \{c_0, c_{12}, c_{13}, c_2, c_3\}$
– $c_i, c_{ij} \in C_i$, $r(c_{ij}) = p_j$, $r(c_0) = p_1$ and $r(c_i) = p_i$ for $i \in \{2, 3\}$.

Notice that the realizability problem for this architecture has already been proven undecidable in [11]. However, we give a different proof of this result here, which can readily be adapted to the case where c_2 and c_3 are not necessarily external output channels but may be read by any other system process. The main idea for the proof given here is the same as in [14]. However, we have to take care of the fact that we have only local specifications at our disposal.

Theorem 20. *The realizability problem is undecidable for regular local specifications for the architecture* $\mathfrak{A}_0$.

Proof. We proceed by a reduction from the halting problem for Turing-Machines, so let $M = (Q, \Sigma, \delta, q_{in}, q_{accept})$ be a deterministic TM and denote $\Sigma_{conf} = \Sigma \cup Q \cup \{\sharp\}$, $\Sigma_{sign} = \{\rightsquigarrow, \curvearrowright\}$. We define the alphabets $\Sigma_{c_0} = \Sigma_{sign_2} \times \Sigma_{sign_3}$ where $\Sigma_{sign_2} = \Sigma_{sign_3} = \Sigma_{sign}$ $\Sigma_{c_{12}} = \Sigma_{c_{13}} = \Sigma_{sign} \times \Sigma_{conf}$ and $\Sigma_{c_2} = \Sigma_{c_3} = \Sigma_{conf}$.

We define the local specifications L_{p_1}, L_{p_2} and L_{p_3} as follows. First, for some local behavior $\alpha = \alpha_0 \frown \alpha_{12} \frown \alpha_{13} \in (\Sigma_{c_0} \times \Sigma_{c_{12}} \times \Sigma_{c_{13}})^\omega$ of process p_1, let $I^j_{\curvearrowright} := \{i \in \mathbb{N} | \Pr_{\Sigma_{sign_j}}(\alpha_0(i)) = \curvearrowright\}$ for $j = 2, 3$ and let $f_d(w) = |\Pr_{\Sigma_{sign_2}}(w)|_{\curvearrowright} - |\Pr_{\Sigma_{sign_3}}(w)|_{\curvearrowright}$ for any prefix $w \sqsubseteq \alpha_0$. Now we define α to be in L_{p_1} if, and only if, *one* of the following conditions does *not* hold

1. for all $i \in \mathbb{N}$ and all $j \in \{2, 3\}$ such that $\Pr_{\Sigma_{conf}}(\alpha_{1l}(i)) \neq \sharp$ we have $pr_{\Sigma_{sign_j}}(\alpha_0(i)) \neq \curvearrowright$
2. for all $i \in \mathbb{N}$ such that $\Pr_{\Sigma_{sign_j}}(\alpha_0(i)) = \curvearrowright$ for some $j \in \{2, 3\}$ we have $\Pr_{\Sigma_{sign_l}}(\alpha_0(i+1)) \neq \curvearrowright$ for all $l \in \{2, 3\}$
3. for all $j \in \{2, 3\}$, $I^j_{\curvearrowright}$ is infinite
4. for all $w \sqsubseteq \alpha_0$ we have $|f_d(w)| \leq 1$

or *all* of the following conditions hold:

5. $\Pr_{\Sigma_{conf}}(\alpha_{12})$ and $\Pr_{\Sigma_{conf}}(\alpha_{13})$ are of the form $\sharp^+ C_0 \sharp^+ C_1 \sharp^+ \ldots$ where each C_i is a configuration of M and $C_0 = q_{in}$ is the initial configuration of M when started on the empty tape
6. for all $i \in \mathbb{N}$ and all $j \in \{2, 3\}$, $\Pr_{\Sigma_{sign}}(\alpha_{1j}(i+1)) = \Pr_{\Sigma_{sign_j}}(\alpha_0(i))$
7. for all $j \in \{2, 3\}$, if $\Pr_{\Sigma_{conf}}(\alpha_{1j}(i)) = \sharp$, then $\Pr_{\Sigma_{conf}}(\alpha_{1j}(l)) = \sharp$ for all $i \leq l \leq \min\{k \in I^j_{\curvearrowright} \,|\, k \geq i\} + 1$
8. for all $j \in \{2, 3\}$ and all $i \in I^j_{\curvearrowright}$, $\Pr_{\Sigma_{conf}}(\alpha_{1j}(i+2)) \neq \sharp$
9. for all $i \in \mathbb{N}$, if $\Pr_{\Sigma_{conf}}(\alpha_{12}(i)) = \Pr_{\Sigma_{conf}}(\alpha_{13}(i)) = \sharp$ and $\Pr_{\Sigma_{conf}}(\alpha_{12}(i+1)) \neq \sharp$ and $\Pr_{\Sigma_{conf}}(\alpha_{13}(i+1)) \neq \sharp$, then $\alpha_{12} = \alpha_{12}{\uparrow_i}\, \sharp C \sharp \alpha'_{12}$ and $\alpha_{13} = \alpha_{13}{\uparrow_i}\, \sharp \widehat{C} \sharp \alpha'_{13}$ and
   – if $f_d(\alpha_0{\uparrow_i}) = 1$, then $\widehat{C} \vdash C$
   – if $f_d(\alpha_0{\uparrow_i}) = -1$, then $C \vdash \widehat{C}$

– if $f_d(\alpha_0{\uparrow_i}) = 0$, then $C = \widehat{C}$

For some local behavior $\alpha = \alpha_{12}{}^\frown\alpha_2 \in (\Sigma^{p_2})^\omega$ of process p_2, we define $\alpha \in L_{p_2}$ if, and only if, $\Pr_{\Sigma_{conf}}(\alpha_{12}(i)) = \Pr_{\Sigma_{conf}}(\alpha_2(i))$. Analogously, for some local behavior $\alpha = \alpha_{13}{}^\frown\alpha_3 \in (\Sigma^{p_3})^\omega$ of process p_3, we define $\alpha \in L_{p_3}$ if, and only if, $\Pr_{\Sigma_{conf}}(\alpha_{13}(i)) = \Pr_{\Sigma_{conf}}(\alpha_3(i))$.

Obviously, L_{p_1}, L_{p_2} and L_{p_3} are all regular. Now we claim that M does not halt on the empty tape if, and only if, p_1, p_2 and p_3 have a joint winning strategy.

First assume that M doesn't halt on the empty tape. Then p_2 and p_3 use the following strategies. They start writing $\sharp$ symbols and they write configurations of the run of M (started on the empty tape) when prompted by the signals $\curvearrowright$ via channel c_{12} or c_{13}, respectively. If they have finished writing a configuration they continue by writing $\sharp$ symbols until they receive the next $\curvearrowright$.

Process p_1 uses the following strategy. It forwards the Σ_{sign_j}-component that it receives on c_0 to p_j for $j \in \{2,3\}$ in the next step. Furthermore, process p_1 writes $\sharp$ symbols to the channels c_{1j} for $j \in \{2,3\}$ until it has to send a $\curvearrowright$ symbol to process p_j. Moreover, each time p_1 sends a $\curvearrowright$-symbol to the channel c_{1j} for some $j \in \{2,3\}$, in the next step, it starts writing the next configuration of the run of M into the same channel. Process p_1 also ends each configuration it has written with a $\sharp$ and writes only $\sharp$ to the channel c_{1j} until it sends the next $\curvearrowright$ to process p_j.

Now let α be a global system behavior which is consistent with the resulting joint strategy of the system processes. Obviously, $\Pr_{\Sigma^{p_2}}(\alpha) \in L_{p_2}$ and $\Pr_{\Sigma^{p_3}}(\alpha) \in L_{p_3}$. Now if one of the conditions 1, 2, 3 and 4 is not fulfilled, then by definition we have $\Pr_{\Sigma^{p_1}}(\alpha) \in L_{p_1}$. If, on the other hand, all these conditions are fulfilled then condition 5 is fulfilled, as, due to condition 3, both processes p_j for $j \in \{2,3\}$ are requested to start a new configuration infinitely many times and, if they are requested to start a new configuration, due to condition 2 they will be given the opportunity to do so. Moreover, each time a process starts to write a configuration, according to condition 1, it will be able to finish this configuration. Conditions 6, 7 and 8 are obviously fulfilled by the definition of the strategy. Finally, if both processes start a new configuration at the same time then, according to condition 4 and the definition of the strategy which writes, successively on demand, the run of M, condition 9 is fulfilled. Hence, $\Pr_{\Sigma^{p_1}}(\alpha) \in L_{p_1}$. Concluding, the joint strategy is winning.

Now assume, conversely, that the machine halts on the empty tape and assume there is a joint winning strategy $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ for the system processes.

First notice that σ_2 cannot take advantage of the information that it receives in the Σ_{conf}-component of c_{12}, that is, if $\alpha_0{}^\frown\alpha_{12}{}^\frown\alpha_2{}^\frown\beta \in (\Sigma_{c_0} \times \Sigma_{c_{12}} \times \Sigma_{c_2} \times \Sigma^{p_3})^\omega$ and $\alpha_0'{}^\frown\alpha_{12}'{}^\frown\alpha_2'{}^\frown\beta' \in (\Sigma_{c_0} \times \Sigma_{c_{12}} \times \Sigma_{c_2} \times \Sigma^{p_3})^\omega$ are two global behaviors which are consistent with σ such that $\Pr_{\Sigma_{sign}}(\alpha_{12}) = \Pr_{\Sigma_{sign}}(\alpha_{12}')$ then $\Pr_{\Sigma_{conf}}(\alpha_{12}) = \Pr_{\Sigma_{conf}}(\alpha_{12}')$ and $\alpha_2 = \alpha_2'$. Analogous for process p_3.

Therefore, as σ_2 uses only the information given in the Σ_{sign}-component of c_{12}, a sequence $i_0, \ldots, i_k \in \mathbb{N} \setminus \{0\}$ for some $k \in \mathbb{N}$ determines the output $\sharp^{i_0}C_0\sharp^{i_1}\ldots\sharp^{i_k}C_k\sharp$ of process p_2 (according to σ_2), in the following way: if process p_2 receives the $\curvearrowright$ symbol for the first time after i_0 steps, σ_2 prescribes to write

some configuration C_0 of M. Then p_2 finishes C_0 by writing a $\sharp$ and if, after additional $i_1 - 1$ steps, p_2 receives the next $\curvearrowright$ symbol then σ_2 again prescribes to write a configuration C_1 of M and so on. Analogous for process p_3.

Now let $k \in \mathbb{N}$ be maximal such that for all sequences $i_0, \ldots, i_k$ the output $\sharp^{i_0} C_0 \sharp^{i_1} \ldots \sharp^{i_k} C_k \sharp$ of process p_2 and the output $\sharp^{i_0} C'_0 \sharp^{i_1} \ldots \sharp^{i_k} C'_k \sharp$ of process p_3 both represent the run of M (up to step k), that means, $C_j \vdash_M C_{j+1}$ and $C'_j \vdash_M C'_{j+1}$ for all $j \in \{0, \ldots, k-1\}$. Notice that such a k exists as both p_2 and p_3 start with the initial configuration of M according to condition 5. Then, for some $l \in \{2, 3\}$ there is a sequence $i_0, \ldots, i_k, i_{k+1}$ such that the output $\sharp^{i_0} C_0 \sharp^{i_1} \ldots \sharp^{i_k} C_k \sharp^{i_{k+1}} C_{k+1} \sharp$ of process p_l does not represent the run of M (up to step $k+1$) and by the choice of k, $C_k \nvdash_M C_{k+1}$. W.l.o.g., let $l = 2$. Now consider the following global system behavior $\alpha = \alpha_0 \frown \alpha_{12} \frown \alpha_{13} \frown \alpha_2 \frown \alpha_3 \in (\Sigma^{\mathfrak{A}_0})^\omega$:

- $\alpha_2 = \sharp^{i_0} C_0 \sharp^{i_1} \ldots \sharp^{i_{k-1}} C_{k-1} \sharp^{i_k} C_k \sharp^{i_{k+1}} C_{k+1} \sharp \beta$ for some $\beta \in \Sigma_{c_2}^\omega$
- $\alpha_3 = \sharp^{i_0} C_0 \sharp^{i_1} \ldots \sharp^{i_{k-1}} C_{k-1} \sharp^{i_k} \sharp^{|C_k|} \sharp^{i_{k+1}} C_k \sharp \beta'$ for some $\beta' \in \Sigma_{c_3}^\omega$
- α_0 is defined by
  $I^j_\curvearrowright = \{n \in \mathbb{N} \mid \alpha_j(n+1) = \sharp \text{ and } \alpha_j(n+2) \neq \sharp\}$ for $j \in \{2, 3\}$

Due to the choice of k, the output of process p_3 which is determined by the sequence $i_0, i_1, \ldots, i_{k-1}, i_k + |C_k| + i_{k+1}$ is $\sharp^{i_0} C_0 \sharp^{i_1} \ldots \sharp^{i_{k-1}} C_{k-1} \sharp^{i_k} \sharp^{|C_k|} \sharp^{i_{k+1}} C_k \sharp$. Therefore, since σ is a winning strategy, given α_0, α_2 and α_3, there is exactly one ω-word $\alpha_{12} \frown \alpha_{13} \in (\Sigma_{c_{12}} \times \Sigma_{c_{13}})^\omega$ such that the resulting global system behavior α is consistent with σ. However, due to condition 9, $\Pr_{\Sigma^{p_1}}(\alpha) \notin L_{p_1}$ as $C_k \nvdash_M C_{k+1}$. □

Theorem 21. *The realizability problem for regular local specifications is undecidable for any architecture with at least three system processes p_1, p_2 and p_3 and channels $c_0 \in C_0$ and $c_{12}, c_{13} \in C_1$ with $r(c_0) = p_1$ and $r(c_{1j}) = j$ for $j \in \{2, 3\}$.*

Proof. Let $\mathfrak{A} = (P, C, r)$ be any architecture with $p_1, p_2, p_3 \in P$, $c_0 \in C_0$, $c_{12}, c_{13} \in C_1$, $r(c_0) = p_1$ and $r(c_{1j}) = j$ for $j \in \{2, 3\}$. First notice that $C_j \neq \emptyset$ for $j = 2, 3$. Now if, $\perp \in r(C_2) \cap r(C_3)$ then Theorem 20 immediately yields undecidability. Moreover, if $r(C_2) \cap (P \setminus \{p_3\} \neq \emptyset$ and $r(C_3) \cap (P \setminus \{p_2\}) \neq \emptyset$ then it is easy to see that the proof of Theorem 20 works for this architecture as well, as process p_1 is better informed than p_2 and p_3 with respect to the external input channel c_0.

Therefore, we only have to consider the case where $r(C_2) = \{p_3\}$ or $r(C_3) = \{p_2\}$. For the proof of this case one can use the same idea as in [5]. We provide the environment with the possibility to send two different encryption functions in each step and we demand of process p_1 that it distributes them to the processes p_2 and p_3. Then, processes p_2 and p_3 have to encrypt their output using the function which they have received in the last step. The set of functions available to the environment can, for example, be chosen as the set of all permutations on Σ_{conf}. Using these encryption functions, the environment can garuantee that processes p_2 and p_3 cannot derive any information from the input that it receives from the other process [5]. Hence, the proof of Theorem 20 can be adapted to the architecture $\mathfrak{A}$. □

Now it easy to see that the previous theorem can be easily extended to Theorem 15: It is not necessary that the environment has a direct channel to the process p_1 but there must be merely a (directed) path from the environment to p_1 which neither contains p_2 nor p_3. This is guaranteed by the assumption that p_1 is reachable and p_2 and p_3 are both not better informed than p_1.

As to the proof of Theorem 16, we first notice that undecidability has been proved in [11] for the following two architectures $\mathfrak{A}_1$ and $\mathfrak{A}_2$.

$\mathfrak{A}_1 = (P, C, r)$ with

- $P = \{p_0, p_1, p_2, p_3\}$
- $C_0 = \{c_{01}, c_{02}\}$, $C_1 = \{c_{13}\}$, $C_2 = \{c_{23}\}$
- $r(c_{0i}) = p_i$ and $r(c_{i3}) = p_3$ for $i \in \{1, 2\}$.

$\mathfrak{A}_2 = (P, C, r)$ with

- $P = \{p_0, p_2, p_2, p_3\}$
- $C_0 = \{c_{01}, c_{02}\}$, $C_1 = \{c_1\}$, $C_2 = \{c_2\}$, $C_3 = \{c_3\}$
- $r(c_{0i}) = p_i$ for $i \in \{1, 2\}$, $r(c_i) = p_{i+1}$ for $i \in \{1, 2\}$ and $r(c_3) = p_3$.

Now, to prove undecidability of $\mathfrak{A}_1$ it is not important which processes read the output channels of process p_3: The process is only needed so that a single (local) specification can talk about both the channels c_{13} and c_{23} at the same time. Then, the original undecidability proof of [14] can be simulated straightforwardly. Moreover, this result can be easily extended to Theorem 16 in the case where p_1 and p_2 are reachable from the environment via two *disjoint* (directed) paths.

If, on the other hand, this is not the case then w.l.o.g. there is a path from p_0 to p_2 which contains p_1 and there is a path from p_2 to p_3. If any path from p_2 to p_3 contains p_1 then we are again in the situation of Theorem 15 so assume that there is at least one such path which does not contain p_1. If, moreover, p_3 has an external output channel then undecidability can be proved just as for the architecture $\mathfrak{A}_2$. Finally, if any output channel of p_3 is read by some other system process then the proof has to adapted with a similar idea as in the proof of Theorem 21.